APPENDIX K

Name Search Suite of Tool(s)

Functional Requirements/Design
Version 1.0

Revised DRAFT
(SNAPIGUI ALPHA VERSION 4)

March 19, 1998

1. General Description

LAS is developing a suite of Name Search tools (i.e., APIs) that can be integrated within an existing customer application or can be used to provide the "guts" of a new customer application. The ***LAS Name Search Suite of Tools*** shall :

- be composed of one or more C++ APIs
- be compatible with any modern platform with a C++ compiler
- provide mechanisms to :
    - compare a query name with one or more candidate names to produce an ordered list of candidate names with the highest probability of representing the same "named" person. This functionality is referred to as the ***Name Comparison Tool(s)*** in the remainder of this document.
    - generate and store intelligent search data for use in extracting relevant subsets of data from large data bases for further evaluation. These mechanisms will facilitate more efficient name searching while ensuring complete and accurate results. This functionality is referred to as the ***Name Extraction Tool(s)*** in the remainder of this document.

The initial offering of the APIs will provide developers with the capability to:

- compare two names to determine the probability that they both represent the same named individual; or
- compare a single query name with a set of candidate names to determine which candidate names are most likely to represent the same named individual.

When a set of candidate names is evaluated, the APIs enable the developer to define the criteria for producing his/her own Results Set. The available options for defining a Result Set include the following:

- an unordered list of all candidate names whose name score exceeds a pre-defined name threshold (e.g., if the threshold = 0, all candidate names will be returned in an unordered list);
- an ordered list of all candidate names whose name score exceeds a pre-defined name threshold (e.g., if the threshold = 0, all candidate names will be returned in an ordered list); or
- an ordered list of the top X candidate names whose name score exceeds a pre-defined name threshold, where X is a number.

1.1 LAS Name Comparison tools

The LAS Name Comparison tools include:

- **NameCheck** - This tool employs multiple evaluation techniques to evaluate and score two names. The NameCheck tool incorporates information regarding variations in spelling, discrepancy in the number of name segments (amount of information included), exclusion of expected information, and positional information in order to establish a name score, which indicates the probability that the two names represent the same individual. The NameCheck tool is controlled by a set of configurable parameters. The NameCheck tool also manages and produces an ordered or unordered list of candidate names with the highest probability of representing the same "named" person, based on the developer-defined criteria for establishing a set of results.

- Various culture-specific tools are available as extensions to the NameCheck tool to perform such functions as the cultural classification of name data (**NameClassifier**), leveling of variations in name data to a single representation (**NameRegularizer**), and the representation of name data based on phonetic similarity (**PhoneticNameKey**).

Version 1.0 of the LAS Name Comparison Tool(s) will establish a baseline of the minimum functionality necessary to perform fuzzy matching on name data. There are two additional enhanced versions of the tool expected to be implemented in-house, prior to producing version 1.0 of a commercially available product. This document defines the functionality to be incorporated into Version 1.0 of the tool, and in some cases, describes why certain decisions were made regarding specific functionality. The document also notes areas for planned future enhancement.

1.2 LAS Name Extraction tools

The LAS Name Extraction tools include:

- **An Intelligent Search Data Generator (ISDG)** which generates one or more search data values that facilitate extraction of relevant information from a data base for further comparative analysis. This tool is a critical component of any search system that must search large volumes of data to locate similar name data. It is not feasible to retrieve and evaluate every name record in a data base to determine its relevance to a query name. The ISDG provides a motivated method for retrieving all relevant information from a data base while reducing the amount of non-relevant information retrieved. This tool can provide significant performance improvements while also ensuring an accurate and complete name search.

- Various culture-specific tools are available as extensions to the ISDG to perform such functions as the cultural classification of name data (**NameClassifier**), leveling of variations in name data to a single representation (**NameRegularizer**), and the representation of name data based on phonetic similarity (**PhoneticNameKey**).

Note that in the current version of this document, there is no further discussion of the Name Extraction Tool(s). These tools will be developed in the future.

2. Perform Error Handling

2.1 Functionality

The tool shall establish a standard list of error codes and their associated text descriptions.

Each function call shall return an error code whenever error checking is appropriate.

The tool shall also provide the capability for the developer to retrieve the text associated with the error code.

The following is a list of the error codes and their meaning:

Error Code	Meaning

3. Produce Linguistic Trace

3.1 Functionality

Version 1.0 will not provide any Linguistic Trace functionality.

4. Accept Input Name Data

4.1 Input Parameters

4.1.1 Functionality

The tool shall verify that all input parameters have valid values as defined in the table below.

The tool shall support several query types (i.e., pre-defined sets of parameters) to facilitate searching the data based on different cultural or other linguistic perspectives.

Certain combinations of these parameters provide better results when addressing known combinations of cultural and/or other linguistic issues.

The tool shall provide the developer with the capability of selecting (defining) one of the API-defined query types.

The tool shall also provide the developer with the capability of modifying any or all of the selected query set parameter values.

At a minimum the tool shall support the following query types :

- Generic;
- Anglo;
- Arabic;
- Chinese;
- Hispanic;
- Korean; and
- Russian.

The tool shall ***not*** allow parameters to be changed in the middle of processing a query. (*see design notes below*).

The set of parameters included in a query type shall include:

Name Field	Parameter	Valid Range / Set of Values	Generic Default Value	Anglo Default Value (Eng/US)	Arabic Default Value (Egypt)	Chinese Default Value (China)	Hispanic Default Value (Mexico)	Korean Default Value (Korea)	Russian Default Value (Russia)
SN	SurnameCheckInitial	{T, F}	F	F	T	F	T	F	T
SN	SurnameCheckVariant	{T, F}	T	T	F	T	T	T	F
SN	SurnameCheckBias	{T, F}	F	F	F	F	F	F	T
SN	SurnameCheckUnknownNotExist	{T, F}	F	F	F	F	F	F	F
SN	SurnameCheckCompressed	{T, F}	F	F	T	F	T	F	F
SN	SurnameAnchorSegment	{first, last, none}	none	none	none	none	first	none	none
SN	SurnameCheckTAQ	{off, remove, score}	score	score	score	score	score	score	score
SN	SurnameMode	{highest, average, lowest}	average	average	average	average	average	average	average
SN	SurnameExactInitialMatchScore	{0.0, 0.1, ...1.0}	1.0	1.0	1.0	0	1.0	0	1.0
SN	SurnameInitialScore	{0.0, 0.1, ...1.0}	0	0	.85	0	.85	0	.85
SN	SNV-Score *	{0.0, 0.1, ...1.0}	-	-	-	-	-	-	-
SN	SurnameOutOfPositionFactor	{0.0, 0.1, ...1.0}	.60	.60	.90	1.0	.60	.63	.80
SN	SurnameAnchorFactor	{0.0, 0.1, ...1.0}	0	0	0	0	.70	0	0
SN	SurnameTAQDisregardAbsentFactor	{0.0, 0.1, ...1.0}	.80	.80	.80	.80	.80	.80	.80
SN	SurnameTAQDeleteAbsentFactor	{0.0, 0.1, ...1.0}	.90	.90	.90	.90	.90	.90	.90
SN	SurnameTAQDeleteFactor	{0.0, 0.1, ...1.0}	.85	.85	.85	.85	.85	.85	.85
SN	SurnameTAQDisregardFactor	{0.0, 0.1, ...1.0}	.70	.70	.70	.70	.70	.70	.70
SN	LastNameUnknownScore	{0.0, 0.1, ...1.0}	.60	.60	.60	.60	.60	.60	.60
SN	NoLastNameScore	{0.0, 0.1, ...1.0}	.65	.65	.65	.65	.65	.65	.65
SN	SurnameCompressedScore	{0.0, 0.1, ...1.0}	.90	.90	.90	.90	.90	.90	.90
SN	SurnameThreshold	{0.0, 0.1, ...1.0}	.50	.50	.63	.70	.60	.63	.62
SN	SurnameWeight	{0.0, 0.1, ...1.0}	1.0	1.0	.80	1.0	1.0	1.0	1.0

GN	GivenNameCheckInitial	{T, F}	T	T	T	F	T	F	T
GN	GivenNameCheckVariant	{T, F}	T	T	F	T	T	T	F
GN	GivenNameCheckBias	{T, F}	F	F	F	F	F	F	F
GN	GivenNameCheckUnknownNotExist	{T, F}	F	F	F	F	F	F	F
GN	GivenNameCheckCompressed	{T, F}	F	F	T	F	T	F	F
GN	GivenNameAnchorSegment	{first, last, none}	none	none	none	none	none	none	first
GN	GivenNameCheckTAQ	{off, remove, score}	score	score	score	score	score	score	score
GN	GivenNameMode	{highest, average, lowest}	average	average	average	lowest	average	average	highest
GN	GivenNameExactInitialMatchScore	{0.0, 0.1, ...1.0}	1.0	1.0	1.0	0	1.0	0	1.0
GN	GivenNameInitialScore	{0.0, 0.1, ...1.0}	.85	.85	.85	0	.85	0	.85
GN	GNV-Score *	{0.0, 0.1, ...1.0}	-	-	-	-	-	-	-
GN	GivenNameOutOfPositionFactor	{0.0, 0.1, ...1.0}	.60	.60	.70	0	.60	.69	.65
GN	GivenNameAnchorFactor	{0.0, 0.1, ...1.0}	0	0	0	0	0	0	.60
GN	GivenNameTAQDisregardAbsentFactor	{0.0, 0.1, ...1.0}	.80	.80	.80	.80	.80	.80	.80
GN	GivenNameTAQDeleteAbsentFactor	{0.0, 0.1, ...1.0}	.90	.90	.90	.90	.90	.90	.90
GN	GivenNameTAQDeleteFactor	{0.0, 0.1, ...1.0}	.85	.85	.85	.85	.85	.85	.85
GN	GivenNameTAQDisregardFactor	{0.0, 0.1, ...1.0}	.70	.70	.70	.70	.70	.70	.70
GN	FirstNameUnknownScore	{0.0, 0.1, ...1.0}	.60	.60	.60	.60	.60	.60	.60
GN	NoFirstNameScore	{0.0, 0.1, ...1.0}	.65	.65	.65	.65	.65	.65	.65
GN	GivenNameCompressedScore	{0.0, 0.1, ...1.0}	.90	.90	.90	.90	.90	.90	.90
GN	GivenNameThreshold	{0.0, 0.1, ...1.0}	.50	.50	.63	.70	.60	.69	.60
GN	GivenNameWeight	{0.0, 0.1, ...1.0}	.80	.80	1.0	.80	.80	.80	.80
SN+GN	NameThreshold **	{0.0, 0.1, ...1.0}	.60	.60	.63	.70	.60	.66	.61

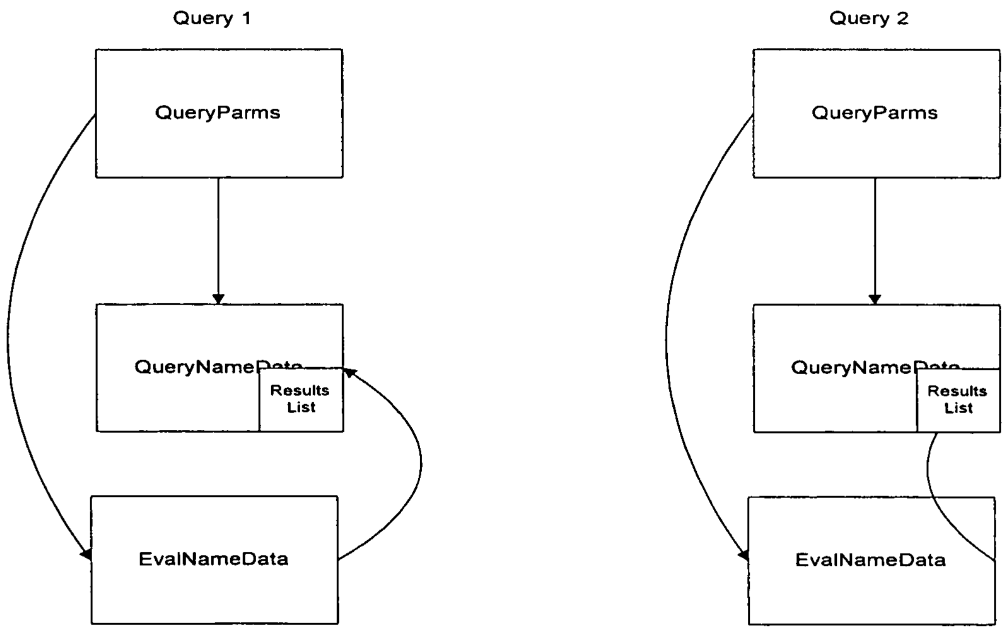
(Note that the values of the parameters that exist in DNC were mapped into their related parameter value in this set of default values. All new parameters were assigned a "best guess" value at this point. Some adjustments were made to the DNC GNOOPS + SNOOPS parameters).

* SNV-Score and GNV-Score values are not included in this table since the scores are actually associated with a specified variant pair, and are contained in the SURNAME-VARIANT and GIVEN-NAME-VARIANT Tables, respectively. The developer can not override the SNV-Score and GNV-Score through the API. Changes to these scores must be made through a separate VariantManager utility. Refer to the sections on SN Variants and GN Variants for more details.

** NameThreshold was calculated by scoring $\frac{(\text{SurnameThreshold}*\text{SurnameWeight})+(\text{GivenNameThreshold}*\text{GivenNameWeight})}{(\text{SurnameWeight}+\text{GivenNameWeight})}$

4.1.2 Design Notes

1. Should the PARMS object be shared or copied to each query-name and evaluation-name object? The following diagram illustrates how all PARMS data would have to be copied and carried with every query object if we are to allow PARMS to be changed in the middle of processing without affecting the queries that are already in progress.. This could result in significant overhead (i.e., memory processing). If the PARMS object is shared then changes to any PARMS in the middle of processing could potentially affect processing that is already in progress. An example of why we would want to change the PARMS in the middle of processing is : We want to re-compare the same query name using the same candidate names but with a different set of parameters. If we do not allow the PARMS to change, then the developer would need to re-call the tool and have the tool re-process the query name and the candidate names in order to compare the names with different parameter settings. The lists of TAQs and Given Name Variants are not considered update-able in the middle of processing. The PARMS that are considered update-able are those PARMS that the developer sets when the tool is called to perform a comparison.



4.1.3 Future Version Notes

1. The tool shall provide the developer with the capability to modify existing or establish new query sets of pre-defined parameters. (Parameter Definition Application).

2.

4.2 Input Name Model

4.2.1 Functionality

The tool shall provide separate function calls for the following name models :

- Given Name + Middle Name + Surname (GMS)
- Given Name + Surname (GS)
- Name (N)
    - Name
    - Surname, Given Name (SN, GN)

The developer will call the desired function and provide the relevant string values in the appropriate function parameters.

The tool shall accept empty string parameter values to the name model function calls. This functionality will be provided to support a customer data base that allows null values or empty strings in any of the fields (e.g., middle name) defined in their name model.

Because the tool itself utilizes the GS model, the most efficient and accurate results will be provided if the GS model is received as input.

4.2.2 Design Notes

1. We selected a function call approach as opposed to passing in a single name string with delimiters as the function call approach will:
    - be easier for a developer to determine which call is appropriate for the business need;
    - not require the developer to identify or understand irrelevant parameters;
    - not require the developer to incorporate irrelevant parameters into their application code; and
    - be more efficient.

4.2.3 Future Version Notes

1. The tool may support the following additional name models/functions :
    - Given Name + Middle Name + Surname + Maiden Name (GMSM)
    - Given Name + Surname + Maiden Name (GSM)

2. The tool may also utilize other name models besides the GS model, if deemed beneficial.

5. Preprocess Name Data

5.1 Functionality

The tool shall preprocess input name data using the following techniques; in the order listed below:

- Identify and parse input name data into given name and surname (name fields)
    - *in future, may use High Frequency Surname data to define surname field*
    - *in future, may move titles and qualifiers into given name field*
    - *in future, unknown and non-existent name values may be used to define given name and surname fields*
- Validate input name data
- Convert name data to UPPER case
    - *in future, this step may move after the TAQ processing (after conjoined TAQ processing is implemented)*
- Preprocess Segmentation and Removal markers (Noise data)
- Parse name fields into name segments
- Identify and process unknown and non-existent name values (e.g., "FNU", "LNU")
- Identify and process minor name parts (e.g., Titles, Affixes, Qualifiers)
    - *in future, identify gender, if applicable*
    - *in future, may identify and process morphological endings separate from TAQs*
- Identify number of segments in name fields
- Identify and process Given Name Variants (Query Only)
    - *in future, identify gender, if applicable*
- Identify and process Surname Variants (Query Only)

5.1.1 Identify and parse input name data into given name and surname (name fields)

5.1.1.1 Functionality

If name data are received in a name model other than GS, then the name data shall be parsed into a GS model.

If the GMS name model is provided, then the internal given name field shall be constructed by placing the input given name in the same field with the input middle name, and the internal surname field shall be set equal to the input surname field.

If the name model does not distinguish the data beyond a single name field (N model), then the tool shall accept the last (i.e., right-most) name segment in the name field as the surname, and place all other name segments in the given name field (e.g., Name: *Jose Garcia Gomez* → Given Name: *Jose Garcia* Surname: *Gomez*). The tool shall recognize the first comma in the Name field (N model) to represent a SN, GN model. The tool shall move the data to the left of the comma into the SN field, and the data to the right of the comma into the GN field (the comma shall be removed from further processing).

The tool shall retain the original form of the parsed Given Name field and Surname field for subsequent processing (i.e., Determine GivenNameCompressedScore, Determine SurnameCompressedScore, and Provide Results information to return to user).

5.1.1.2 Future Version Notes

1. The tool may move all Titles, and Qualifiers into the Given Name field.
2. The tool may utilize High Frequency (HF) Surname and TAQ data to determine the structure of the name data (e.g., two HF Hispanic Surnames found in a Name string can be used to identify a Hispanic Surname; ABU means "father of" and BIN means "son of" in Arabic names. Reference the example above, Name: "Jose Garcia Gomez" → Given Name: "Jose" Surname: "Garcia Gomez").
3. With Arabic names, the tool may move all name segments other than the first name segment (presumably the first given name) into the SN field – clearly this functionality can not be implemented until the NameClassifier tool is made available. Other criteria may be used such as TAQ values.
4. If HF Surnames are found anywhere in the GN field, the tool may move them to the SN field. This may prove beneficial to handling multi-segment surnames such as those that occur in the Hispanic naming system. Sometimes HF Surnames appear in the GN field because they are aliases, so we must be careful with this.
5. The tool may utilize "NFN", "NMN", and "NLN" values when creating the name fields. The only contents of the SN field should be "NLN" or "LNU" if they occur anywhere in the name. If "NFN", "FNU", "MNU", or "NMN" occur, then they should occur only in the GN field. However, additional values may be allowed in the GN field. If name models other than the GS model are utilized within the tool itself, the tool may support more sophisticated processing of these values (e.g., the GMS)
6. Since we decided that we would not handle Maiden name data at this time, there is no special handling of middle name data at this time. Future versions of the tool may use gender data, if available, to manipulate the middle name when dealing with female data – only when dealing with Anglo names, however.

5.1.2 Validate input name data

5.1.2.1 Functionality

The tool shall assume that all name data are person names.

The tool shall accept name data (given name plus middle name plus surname) up to 255 character length total. Thus, the tool shall support Given Name <= 255 characters, MN <= 255 characters, and

Surname <= 255 characters, since any one of these fields may contain all of the name data. The tool shall truncate any data that exceeds the specified maximum character length.

The tool shall accept any case as input (i.e., upper, lower, or mixed case).

The tool expects roman characters (i.e., alphabetic, numeric, and punctuation markers) as input, however it shall accept both roman and non-roman characters and process them in the same manner.

Through Version 1.0, the tool shall not support double-byte character sets.

5.1.2.2 Future Version Notes

1. The tool may recognize and support double-byte character sets (e.g., unicode).

5.1.3 Convert name data to UPPER case

5.1.3.1 Functionality

The tool shall change all name data to upper case.

5.1.3.2 Design Notes

5.1.3.3 Future Version Notes

1. The tool may reference the TAQ Table to process the name data more intelligently prior to converting the data from mixed case to upper case. TAQ data may facilitate more intelligent segmentation of the name data (e.g., "VanDerMinten" → "Van Der Minten" or "DAngelo" → "D Angelo"). The tool may look up the values "Van", "Der", and "Minten" in the TAQ Table; if found, then the tool shall identify the values as TAQs and process them appropriately. If not found, the tool shall rejoin any remaining non-TAQs and then convert them to upper case.

2. The tool may also reference a High Frequency Surname Table to process the name data more intelligently prior to converting the data from mixed case to upper case. High Frequency Surname data may facilitate more intelligent segmentation of the name data (e.g., "GarciaGomez" → "Garcia Gomez"). The tool may look up the Surnames "Garcia" and "Gomez" in the High Frequency Surname Table; if found, then the tool shall identify the Surnames as High Frequency Surnames and process them appropriately. If not found, the tool shall rejoin any remaining non-High Frequency Surnames and then convert them to upper case.

3. The tool may utilize case information to process the name data more intelligently in conjunction with TAQ and HF Surname processing (e.g., DeLaCruz → De La Cruz or GarciaGomez → Garcia Gomez) to assist in determining and parsing name segments.

5.1.4 Preprocess Segmentation and Removal markers (Noise data)

5.1.4.1 Functionality

The tool shall recognize non-alphabetic characters received in the name model.

The tool shall reference a list of replaceable single character non-alphabetic data values to determine what process if any is required for the character encountered.

The tool shall replace each item identified in the list of single character values (i.e., punctuation and/or numbers) with their designated single character REPLACEMENT VALUE as identified in the replacement list.

The tool shall only allow a marker to be defined as either a removal marker or a segmentation marker. If a marker is defined as both a removal marker and a segmentation marker, then the tool shall recognize the marker as a removal marker.

The Table below illustrates the default contents of the replacement list.

Note that the REPLACEMENT-VALUE in the table below is a textual description of an empty string (designated by NIL) or a space (designated by BLANK), which is provided for ease of reading, and is not necessarily representative of the physical representation of the list referenced by the tool.

The tool shall recognize a list of single character markers that indicate the end of a name segment / beginning of a new name segment by replacing the segmentation markers with a space (designated by BLANK in the table below).

The tool shall recognize standard segmentation delimiters such as tab, new line, carriage return, etc. without them being explicitly entered into the segmentation list.

The tool shall recognize a list of markers that are designated for removal by deleting the values entirely from the name field (i.e., mapping each removal value to an empty value or no value; designated by NIL in the table below).

The tool shall provide default lists of removal markers and segmentation markers.

The tool shall allow the developer to provide a custom removal list.

The tool shall also allow the developer to provide a custom segmentation list.

The tool shall accept an empty removal list to indicate turning off removal processing. If a BLANK is included in the removal list, multiple segment name fields shall be recognized by the tool as a single segment.

The tool shall accept an empty segmentation list to indicate turning off segmentation processing. If an empty segmentation list is provided, multiple segment name fields shall be recognized by the tool as a single segment.

If the developer does not provide a segmentation list at all, the tool shall utilize its own default segmentation list.

If the developer does not provide a removal list at all, the tool shall utilize its own default removal list.

VALUE	REPLACEMENT VALUE
BLANK	BLANK (segmentation value)
!	NIL (removal value)
"	NIL (removal value)
#	NIL (removal value)
$	NIL (removal value)
%	NIL (removal value)
(	NIL (removal value)
)	NIL (removal value)
*	NIL (removal value)
+	NIL (removal value)
-	BLANK (segmentation value)
.	NIL (removal value)
/	NIL (removal value)
:	NIL (removal value)
;	NIL (removal value)
<	NIL (removal value)
=	NIL (removal value)
>	NIL (removal value)
?	NIL (removal value)
@	NIL (removal value)
[	NIL (removal value)
\	NIL (removal value)
]	NIL (removal value)
`	NIL (removal value)
{	NIL (removal value)
|	BLANK (segmentation value)
}	NIL (removal value)
'	NIL (removal value)
0	NIL (removal value)
1	NIL (removal value)
2	NIL (removal value)
3	NIL (removal value)
4	NIL (removal value)
5	NIL (removal value)
6	NIL (removal value)
7	NIL (removal value)

8	NIL (removal value)
9	NIL (removal value)

5.1.4.2 Design Notes

1. We decided not to implement a string translation Table or regular expressions at this time for three primary reasons:

    - We determined that it would be easier to implement single character replacements at this time.

    - We were not able to determine that there really is a need at this point to handle more than single character replacements, as rewrite rules and other techniques such as the TAQ processing (separate-if-conjoined, disregard, and delete) satisfied all of the examples we could come up with for string replacements. The example of differences in handling apostrophes was resolved by TAQ processing.

    - Although regular expressions may provide the most flexible solution, our current regular expression routines are restricted to the windows environment and we will need more time to investigate alternative approaches prior to their implementation in the tool.

2. Even so, in the future, the tool may pre-process punctuation through a string replacement Table or through regular expressions to enable us to replace contextualized punctuation with some string value, if necessary. These preprocessing rules will probably be culture-specific, and therefore, will also require that the tool support culture-specific processing.

3. Special non-roman characters that often appear intermixed with roman characters, (e.g.., □, □, □, □), will probably be handled with rewrite rules via regular expressions by other functions yet to be defined for the tool.

5.1.4.3 Future Version Notes

1. We may want to look further at " " and ( ) because these values are sometimes used to designate an alias or nickname when they appear in either the SN field or the GN field (e.g., PITRA **"PETROFF"**, SANTANA ANDRE or EMAN, JAN **(HENNY)** H.).

5.1.5 Parse name fields into name segments

5.1.5.1 Functionality

Name fields shall be parsed into name segments with remaining punctuation in tact (i.e., any punctuation not processed in 2.4 shall be left in tact in the name data).

The tool shall define a name segment as a string of text surrounded by white space.

The tool shall support up to 10 segments in both the SN and GN fields prior to removal of TAQs.

If more than 10 segments are provided in Version 1.0, any additional segments will simply be excluded from the evaluation process.

The tool shall support name segments up to 30 characters in length.

5.1.5.2 Future Version Notes

1. The tool may segment two HF names if found conjoined.

5.1.6 Identify and process unknown and non-existent name values

5.1.6.1 Functionality

The tool shall recognize the following special characters which indicate unknown or non-existent name segment values:

- "FNU" - representing first name unknown;
- "MNU" - representing middle name unknown;
- "LNU" - representing last name unknown;
- "NFN" - representing no first name;
- "NMN" - representing no middle name; and
- "NLN" - representing no last name.

The tool shall replace any GN segment containing "FNU", "MNU", "NFN", and "NMN" with an empty GN segment and tag the segment as "unknown" or "not-exist".

The tool shall replace any SN segment containing "LNU" and "NLN" with an empty SN segment and tag the segment as "unknown" or "not-exist".

The tool shall tag all other SN or GN segments as "known".

5.1.6.2 Design Notes

1. Tagging these special values will enable the tool to use this information during the evaluation process.

2. The tool assumes that the name field processing has already addressed the issue that "LNU" and "NLN" should not appear in the GN field and that "FNU", "MNU", "NFN", and "NMN" should not appear in the SN field. Thus, there is no special handling done at this point in the process.

5.1.7 Identify and process minor name parts (e.g., Titles, Affixes, Qualifiers)

5.1.7.1 Functionality

The tool shall identify Titles, Affixes (prefixes, suffixes, infixes), and Qualifiers (TAQs) in the name data, by referencing a Table of single segment TAQs (i.e., no complex TAQs, such as *al din*) that are defined within the context of a specified cultural group or partition. Note that the TAQ Table does not include punctuation in Version 1.0. Future versions may include punctuation such as *O'*.

5.1.7.1.1 TAQ Table

The TAQ Table shall contain Titles, Affixes, and Qualifiers that are described in context of a specified cultural group or partition (i.e., CULT-AFF-ID). In Version 1.0 of the tool, the TAQ table shall include a "Generic" partition (i.e., non-culture specific), as well as Anglo, Arabic, Chinese, Hispanic, Korean, and Russian. The table below lists the cultural partitions that will be included in Version 1.0:

CULT-AFF-ID	CULTURAL-AFFINITY
A	Arabic
C	Chinese
E	Anglo
G	Generic
H	Hispanic
K	Korean
R	Russian

A "Generic" partition shall be composed of the most commonly occurring TAQ values that can be evaluated as a TAQ value regardless which culture is being evaluated. In other words, "Generic" TAQs frequently occur in multiple cultures. For example, "MR" and "PHD" often occur in a variety of multi-cultural names. Titles and Qualifiers readily fall into the "Generic" category. Affixes are less likely to occur in the "Generic" category.

In most cases, TAQ values that are included in the "Generic" partition will not be included in a cultural partition, even though they may be associated with a specific culture. Exceptions will occur when the TAQ definition (TAQ-TYPE-CODE, GENDER, SEPARATE-IF-CONJOINED, SN-PROCESS-ID, or GN-PROCESS-ID) or TAQ processing is distinct in the different cultures. For example, "SR" is an abbreviation of the Hispanic title "Senor" as well as a Generic qualifier indicating "senior or the first".

The TAQ Table shall not be modifiable by the developer or user in Version 1.0.

A separate data base utility will be developed to generate code representing the contents of the TAQ Table which is currently stored in MS Access.

An example of the contents of the TAQ Table is provided below:

CULT-AFF-ID	TAQ	TAQ-TYPE-CODE	SEPARATE-IF-CONJOINED	GENDER	SN-PROCESS-ID	GN-PROCESS-ID
A	AABD	P	0	U	DIS	DIS
A	AAL	P	0	U	DIS	DIS
A	ABA	P	0	M	DIS	DIS
A	ABBD	P	0	U	DIS	DIS
G	ABD	P	0	U	DIS	DIS
A	ABDAL	P	0	U	DIS	DIS
A	ABDAN	P	1	U	DIS	DIS
A	ABDAR	P	1	U	DIS	DIS
A	ABDAS	P	0	U	DIS	DIS
G	ABDEL	P	0	U	DIS	DIS
A	ABDEN	P	1	U	DIS	DIS
A	ABDER	P	1	U	DIS	DIS
A	ABDES	P	1	U	DIS	DIS

Each TAQ shall be classified as a P-Prefix, S-Suffix, T-Title, I-Infix, Q-Qualifier (TAQ-TYPE-CODE). *Note that at the present time, we have no Infixes in the Table.*

Each TAQ shall be classified as to whether or not the TAQ shall be recognized and separated from a stem, if the TAQ occurs conjoined with the stem (SEPARATE-IF-CONJOINED, 1="T", 0="F"). Version 1 of the tool shall not process SEPARATE-IF-CONJOINED.

Each TAQ shall be assigned a gender (GENDER) of either "M" - male, "F" - Female, or "U" - Undefined. Version 1 of the tool shall not process GENDER.

Each TAQ shall be classified distinctly for the SN field (SN-PROCESS-ID) and GN field (GN-PROCESS-ID) as either a DELETE TAQ ("DEL") or a DISREGARD TAQ ("DIS").

5.1.7.1.2 TAQ Processing

If the parameter GivenNameCheckTAQ = "off", the tool shall not perform any TAQ processing in the GN field.

If the parameter SurnameCheckTAQ = "off", the tool shall not perform any TAQ processing in the SN field.

If the parameter GivenNameCheckTAQ = "remove" or "score", the tool shall perform TAQ processing in the GN field as described below.

If the parameter SurnameCheckTAQ = "remove" or "score", the tool shall perform TAQ processing in the SN field as described below.

The tool shall consider TAQs to occur anywhere within the GN field or GN segment, if the GivenNameCheckTAQ = "remove" or "score".

The tool shall consider TAQs to occur anywhere within the SN field or SN segment, if the SurnameCheckTAQ = "remove" or "score".

The tool shall recognize and remove all TAQs from the GN and SN name fields, but retain the actual value of the TAQ as well as the classification of the TAQ as a DELETE TAQ or DISREGARD TAQ. Each TAQ's set of retained information shall be associated with the TAQ's related name segment for use in the evaluation process.

The tool shall recognize TAQs via the following steps:

- First, the tool shall look up each GN or SN segment to determine whether it is included in the TAQ Table for the appropriate cultural perspective (CULT-AFF-ID) as defined by the API-defined query type.
    - If the GN or SN segment is included in the subset of the TAQ Table associated with the selected cultural perspective, the tool shall process the TAQ according to the culture-specific definition, as described below.
    - If the GN or SN segment is not included in the subset of TAQ Table associated with the selected cultural perspective, and the selected cultural perspective is not "Generic", then the tool shall look up each GN or SN segment to determine whether it is included in the "Generic" subset of the TAQ Table.
        - If the GN or SN segment is included in the "Generic" subset of the TAQ Table, the tool shall process the TAQ according to the culture-specific definition, as described below.
        - If the GN or SN segment is not included in the "Generic" subset of the TAQ Table, the tool shall perform no additional TAQ processing for this GN or SN segment.

The tool shall utilize the culture-specific definition of the TAQ (information in the TAQ Table about each TAQ) to determine its related segment in the following manner:

- Stems are defined as any segment whose value is not defined as a TAQ (i.e., is not included in the TAQ Table);
- Any TAQ located to the left of the first stem will be associated with the first stem;
- Any TAQ located to the right of the final stem will be associated with the final stem; and
- For medial TAQs, the following rules shall apply:
    - Find the rightmost suffix (as defined in the TAQ Table) following a stem;

- That suffix and any other TAQ preceding it shall be associated with the preceding stem; and
- Any remaining TAQs shall be associated with the next stem.

The following example illustrates how TAQs will be associated with stem segments.

DOKTOR	ABD	EL	**RAHMAN**	**NOOR**	EL	DIN	ABD	EL	**KADIR**
T1	P1	P2	STEM1	STEM2	P3	S1	P4	P5	STEM3

T1 P1 P2 are all associated with STEM1
P3 S1 are associated with STEM2
P4 P5 are associated with STEM3

If every name segment contained in a name field is identified as a TAQ, then the tool shall associate all of the TAQs with a single empty segment.

TAQs may occur conjoined with a name stem, as is the case with DeLaCruz, O'Connor, and MacDougal, or they can occur as disjoined segments within a name, as in De La Cruz, O' Connor and Mac Dougal. *Version 1.0 of the tool shall not recognize or process conjoined TAQs.*

5.1.7.2 Design Notes

1. A selected subset of the current corporate TAQ Table will be designated for inclusion in the product TAQ Table.
2. The tool will not recognize complex TAQs, such as "al din". Previous prototypes for the State Department (legacy ANA) have supported complex TAQs for reasons that are not relevant to this tool. For more information on this issue, refer to the corporate linguistic data repository TAQ documentation.
3. The tool will accept empty strings in the SN and GN fields, so there is no special processing to handle the situation when only TAQ(s) occur in one of the name fields.

5.1.7.3 Future Version Notes

1. The developer or user shall be provided mechanisms for adding new TAQs to the TAQ Table.
2. The developer or user shall not be allowed to delete TAQs from the TAQ Table.
3. The need for DELETE and DISREGARD tags for TAQs may be eliminated – DELETE and DISREGARD relations may both be replaced by a single matrix of relationships between

TAQs (as described in the Apply Surname TAQ Factors and Apply Given Name TAQ Factors sections of this document).

4. If DELETE and DISREGARD tags are maintained distinctly, the tool may also support DELETE and DISREGARD processing of Infixes, if necessary. Currently there are no infixes defined in the TAQ Table.

5. The tool may support conjoined TAQs.

    - Conjoined TAQs can be very effective in dealing with morphological endings.... (i.e., conjoined suffixes such as -son, -man, -ovich). Conjoined suffixes may be supported prior to cultural specific handling. Morphological endings trigger left bias right now, so we will need to consider left bias when implementing morphological lookups either as part of TAQ processing, or independent of it.

    - Conjoined TAQs will be even more effective if the tool supports culture-specific processing.

    - Once culture-specific processing is supported, the tool shall recognize that all TAQ types can be conjoined with a stem (i.e., prefix, suffix, infix, title, qualifier).

    - If a TAQ is identified as conjoined (i.e., the field SEPARATE-IF-CONJOINED = "T", then the tool shall consider this TAQ if it is conjoined to a stem as well as if it is a stand-alone name segment (i.e., the TAQ is surrounded by white space); if the tool is identified as not conjoined (i.e., the field SEPARATE-IF-CONJOINED = "F",), then the tool shall consider the TAQ if found as a stand-alone name segment. Thus, the SEPARATE-IF-CONJOINED field indicates whether the application program will search for a TAQ as an independent name segment as well as part of a name segment.

    - Conjoined processing shall determine whether a TAQ is conjoined either at the beginning or at the end of a name segment. Conjoined processing does <u>not</u> search for a TAQ anywhere within the name segment.

    - If the tool identifies a conjoined TAQ in a name segment, it shall:

        - create multiple segments by separating the TAQ(s) from the stem; and

        - then proceed with TAQ processing of the separated segments.

6. There is an outstanding issue for handling the apostrophe – this issue will not be resolved in Version 1.0. The issue is that in some cases, such as with a name like "O'Connor", we want to separate the conjoined "O' "from "Connor", and then recognize the "O' " as a TAQ and process it as is defined in the TAQ Table (i.e., DISREGARD or DELETE). In a case such as "Ol'ga", however, we simply want to delete the apostrophe to produce "Olga". For Version 1.0, we are defining the apostrophe as a removal marker and mapping its

occurrence to "NIL"; so "Oconnor" and "Olga" will be produced for the examples cited above. Once the tool supports conjoined TAQs, the TAQ "O" may be recognized and processed as a conjoined TAQ, which would produce the desired "Connor". The current TAQ Table has entries for "D' ", 'L' ", and "O' " as well as their counterparts, "D", "L", and "O" – all of which are marked SEPARATE-IF-CONJOINED. Further analysis is required to determine whether it is necessary to handle the apostrophe using a different technique(s). If we determine that the apostrophe will always (for all foreseeable future versions of the tool) be removed prior to TAQ processing, then the "D' ", 'L' ", and "O' " entries in the TAQ Table will no longer be necessary.

7. The tool may attempt to define the gender of a name based on the available TAQs.
8. In later versions of the tool, additional cultural partitions may be supported in the TAQ Table.
9. In the future, the tool may process TAQs differently based on culture (note that this is part of the justification for separating Generic from Anglo).

5.1.8 Identify number of segments in name fields

5.1.8.1 Functionality

After TAQ removal, the system shall identify the number of segments in the SN and GN fields to assist in producing the ordered list of the top X names.

At a minimum, the tool shall require one SN segment and one GN segment for each name.

The tool shall accept an empty string as a single SN segment or GN segment.

If no segment exists after TAQ removal, the tool shall create a single empty segment for the appropriate name field. Thus, the tool shall recognize a single empty segment to indicate no data after TAQ removal.

The tool shall tag all empty segments as "unknown".

The tool shall support up to 5 segments in both the SN and GN fields after removal of TAQs.

If more than 5 segments remain in a name field after TAQ removal, the additional segments will be excluded from the evaluation process.

5.1.9 Identify and process Given Name Variants (Query Name Only)

5.1.9.1 Functionality

5.1.9.1.1 GIVEN-NAME-VARIANT Table

The tool shall support a single GIVEN-NAME-VARIANT Table that describes the relationship between two Given Names based on a specified cultural perspective (GN-CULT-AFF-ID).

The GIVEN-NAME-VARIANT Table will consist of pairs of given names *within a culture* that are determined to be variants of one another, based on their having the same name stem. In other words, the type of variation defined for the contents of the GIVEN-NAME-VARIANT Table are determined based on a specific cultural perspective (e.g., using an english or anglo perspective, "ELENA" and "HELENA" are considered "similar but different" names", however, they are considered "predictable spelling variants" when the pair is defined using a Hispanic perspective).

The criteria for whether or not a pair of variants will be included in the GIVEN-NAME-VARIANT Table will be based on the following defined types of variation:

Variation Values

VARIATION TYPE	EXAMPLE	DEFAULT VALUE
*Spelling variant - predictable	SEAN - SHAWN	0.95
*Abbreviation	MARIA - MA	0.90
*Nickname	FRANCISCO - PACO	0.90
*Same root - morph difference	BUSTO - BUSTOS	0.85
Different culture (translation)	FRANCISCO - FRANCIS	0.85
*Related - unpredictable difference	BUSTO - BUSTONES	0.80
Truncation	FRANCISCO - FRANCISC	0.70
Misspelling	MARIA - MRAIA	0.70
Similar name; not same root	SALAM - SALIM	0.65
Gender	MARIA - MARIO	0.50

The items marked with a * are culture-specific variants:

- **Spelling variation** may or may not be taken care of by digraph matching (for the product it will probably handle most reasonable variation).
- **Abbreviations** and **nicknames** depend on the culture; many, if not most, can be taken care of with lists that can be improved over time by restricting the culture relationship.
- **Same root/morphological difference** is definitely culture-specific, since the root and morphological elements can only be identified within a system; many of the differences (if they are short) can be handled with digraphs.

- **Related/unpredictable difference** is also within a cultural system; these are not the same name, however. Much of these differences can be handled with digraphs, too.
- **Truncation** and **misspelling** can also be said to be culture-specific, since you have to know how it was spelled in the first place to know if it's misspelled or truncated. Depending on how these are identified (i.e., if we know for sure what they are a variant of), these should perhaps receive a high value (e.g., 0.90).

Similar name; not same root variants will be included in the GIVEN-NAME-VARIANT Table to enable the tool to override a potentially high digraph score by assigning a lower variant score, if desired. Thus, a name that might qualify as a digraph variant, but which we do not consider a variant of the related name pair, would be less likely to qualify as a variant (e.g., "MUHAMAD" and "MAHMUD" may occur in the Table with an associated GNV-SCORE set very low because their variation type = "similar but different" and we would prefer not to see them appear as variants of one another).

In Version 1.0 of the tool, the CULT-AFF-ID will include Generic, Anglo, Arabic, Chinese, Hispanic, Korean, and Russian, if applicable. The table below lists the possible CULT-AFF-ID values:

CULT-AFF-ID	CULTURAL-AFFINITY
A	Arabic
C	Chinese
E	Anglo
G	Generic
H	Hispanic
K	Korean
R	Russian

The GIVEN-NAME-VARIANT Table shall not be modifiable by the developer or user in Version 1.0.

A separate data base utility will be developed to generate code representing the contents of the GIVEN-NAME-VARIANT Table which is currently stored in MS Access.

The following is a sample of the contents of the GIVEN-NAME-VARIANT Table:

GIVEN-NAME	GN-VARIANT	GNV-SCORE	GN-CULT-AFF-ID
AARON	AHARON	0.95	G
AARON	ARN	0.65	G
ABRAHAM	ABE	0.9	G
ABRAHAM	ABRAM	0.65	G
ABRAHAM	AVRAHAM	0.95	G
ABRAHAM	AVROM	0.65	G
ADAM	ADAMO	0.85	G
ADAM	ADAN	0.85	G
ADRIAN	ADRIEN	0.95	G
AGNES	AGGIE	0.9	G
AGNES	AGNESE	0.85	G

AGNES	INES	0.85	G
AHMED	AHMAD	0.95	G
ALAN	AL	0.9	G

Each entry in the GIVEN-NAME-VARIANT Table shall represent a bilateral relationship, and therefore only one entry will be required to support these bilateral relations (e.g., there will only be one entry in the table to define a relationship between "ABRAHAM" and "ABE").

Each entry in the GIVEN-NAME-VARIANT Table will be assigned a GNV-SCORE, which is based on a type of variation. The variation type will not be included in the GIVEN-NAME-VARIANT Table.

The GIVEN-NAME-VARIANT Table will not include self-relationships (i.e., "ABRAHAM" "ABRAHAM" 0 is not in the Table).

5.1.9.1.2 Given Name Variant Processing

If GivenNameCheckVariant = "T", the tool shall perform the following:

- First, the tool shall look up each query GN segment to determine whether it is included in the GIVEN-NAME-VARIANT Table for the appropriate cultural perspective (GN-CULT-AFF-ID) as defined by the API-defined query type.
    - If the query GN segment is included in the subset of the GIVEN-NAME-VARIANT Table associated with the selected cultural perspective, the tool shall associate all of its known variants within that cultural perspective, and their variation score (GNV-SCORE) with the query GN segment for use in the evaluation process.
    - If the query GN segment is not included in the subset of the GIVEN-NAME-VARIANT Table associated with the selected cultural perspective, and the selected cultural perspective is not "Generic", then the tool shall look up each query GN segment to determine whether it is included in the "Generic" subset of the GIVEN-NAME-VARIANT Table.
        - If the query GN segment is included in the "Generic" subset of the GIVEN-NAME-VARIANT Table, the tool shall associate all of its known variants within the "Generic" subset, and their variation score (GNV-SCORE) with the query GN segment for use in the evaluation process.
        - If the query GN segment is not included in the "Generic" subset of the GIVEN-NAME-VARIANT Table, the tool shall perform no additional Given Name Variant processing for this GN segment.

5.1.9.2 Design Notes

1. In version 1.0, the Anglo contents of the GIVEN-NAME-VARIANT Table are derived from the 389 given names that occurred at least 200 times in the State Department Passport Database.

2. Hispanic Given Name Variants were gathered from the HNA variant table LAS generated for the State Department.

3. Korean Given Name Variants were gathered from the data LAS generated for ORD-C.

4. Chinese Given Name Variants were gathered from the DNC variant table LAS generated for the State Department.

5. Arabic Given Name Variants were gathered from the DNC variant table LAS generated for the State Department.

5.1.9.3 Future Version Notes

1. The developer or user shall be provided mechanisms for adding new variants to the GIVEN-NAME-VARIANT Table.

2. The developer or user shall not be allowed to delete variants from the GIVEN-NAME-VARIANT Table.

3. The tool may attempt to determine the gender of a name based on the Given Name variants – this may be especially valuable for Hispanic given names. In order to do this, the GIVEN-NAME-VARIANT Table would be enhanced to include gender information.

4. The GIVEN-NAME-VARIANT Table may support additional cultural perspectives.

5. In the future, the tool may process GN Variants differently based on culture (note that this is part of the justification for separating Generic from Anglo).

5.1.10 Identify and process Surname Variants (Query Name Only)

5.1.10.1 Functionality

5.1.10.1.1 SURNAME-VARIANT Table

The tool shall support a single SURNAME-VARIANT Table that describes the relationship between two Surnames based on a specified cultural perspective (SN-CULT-AFF-ID).

The SURNAME-VARIANT Table will consist of pairs of surnames *within a culture* that are determined to be variants of one another, based on their having the same name stem. In other words, the type of variation defined for the contents of the SURNAME-VARIANT Table are determined based on a specific cultural perspective.

The criteria for whether or not a pair of variants will be included in the SURNAME-VARIANT Table will be based on the following defined types of variation:

Variation Values

VARIATION TYPE	EXAMPLE	DEFAULT VALUE
*Spelling variant - predictable	GOMEZ - GOMES	0.95
*Abbreviation	GOMEZ - GOM	0.90
*Same root - morph difference	BUSTO - BUSTOS	0.85
*Related - unpredictable difference	BUSTO - BUSTONES	0.80
Truncation	FRANCISCO - FRANCISC	0.70
Misspelling	GOMEZ - GMEZ	0.70
Similar name; not same root	GOMEZ - GAMEZ	0.65

The items marked with a * are culture-specific variants:

- **Spelling variation** may or may not be taken care of by digraph matching (for the product it will probably handle most reasonable variation).
- **Abbreviations** and **nicknames** depend on the culture; many, if not most, can be taken care of with lists that can be improved over time by restricting the culture relationship.
- **Same root/morphological difference** is definitely culture-specific, since the root and morphological elements can only be identified within a system; many of the differences (if they are short) can be handled with digraphs.
- **Related/unpredictable difference** is also within a cultural system; these are not the same name, however. Much of these differences can be handled with digraphs, too.
- **Truncation** and **misspelling** can also be said to be culture-specific, since you have to know how it was spelled in the first place to know if it's misspelled or truncated. Depending on how these are identified (i.e., if we know for sure what they are a variant of), these should perhaps receive a high value (e.g., 0.90).

Variant Surnames based on morphological endings will only be included, if they will not be handled by other processing (i.e., conjoined TAQ (i.e., suffix) removal processing or by a morphological lookup table that will trigger the left bias factor). In other words, we are focusing on stem variations.

Similar name; not same root variants will be included in the SURNAME-VARIANT Table to enable the tool to override a potentially high digraph score by assigning a lower variant score, if desired. Thus, a name that might qualify as a digraph variant, but which we do not consider a variant of the related name pair, would be less likely to qualify as a variant.

The following additional types of variation will <u>not</u> be included in the SURNAME-VARIANT Table (even though they are included in the GIVEN-NAME-VARIANT Table):

- Nicknames;
- Different culture (translation); and
- Gender variants.

In Version 1.0 of the tool, the CULT-AFF-ID will include Generic, Anglo, Arabic, Chinese, Hispanic, Korean, and Russian, if applicable. The table below lists the possible CULT-AFF-ID values:

CULT-AFF-ID	CULTURAL-AFFINITY
A	Arabic
C	Chinese
E	Anglo
G	Generic
H	Hispanic
K	Korean
R	Russian

The SURNAME-VARIANT Table shall not be modifiable by the developer or user in Version 1.0.

A separate data base utility will be developed to generate code representing the contents of the SURNAME-VARIANT Table which is currently stored in MS Access.

The following is a sample of the contents of the SURNAME-VARIANT Table:

SURNAME	SN-VARIANT	SNV-SCORE	SN-CULT-AFF-ID
ACOSTA	COSTA	0.85	H
AGUILAR	AGUILA	0.65	H
AGUILAR	AGUILERA	0.65	H
AGUILERA	AGUILA	0.85	H
AGUILERA	AGUILAR	0.65	H
ALBA	ALBAN	0.65	H
ALCANTARA	ALCANTAR	0.85	H
ALDANA	ALDAMA	0.8	H
ALMANZAR	ALMANZA	0.65	H
ALONSO	ALONZO	0.95	H
ALONZO	ALONSO	0.95	H
ALVARADO	ALVARDO	0.8	H
ALVAREZ	ALVARES	0.95	H
ALVAREZ	ALVARO	0.65	H
ALVAREZ	ALVEREZ	0.8	H

Each entry in the SURNAME-VARIANT Table shall represent a bilateral relationship, and therefore only one entry will be required to support these bilateral relations (e.g., there will only be one entry in the table to define a relationship between "GOMEZ" and "GOMES").

Each entry in the SURNAME-VARIANT Table will be assigned a SNV-SCORE, which is based on a type of variation. The variation type will not be included in the SURNAME -VARIANT Table.

The SURNAME-VARIANT Table will not include self-relationships (i.e., "GOMEZ" "GOMEZ" 0 is not in the table).

5.1.10.1.2 Surname Variant Processing

If SurnameCheckVariant = "T", the tool shall perform the following:

- First, the tool shall look up each query SN segment to determine whether it is included in the SURNAME-VARIANT Table for the appropriate cultural perspective (SN-CULT-AFF-ID), as defined by the API-defined query type.
    - If the query SN segment is included in the subset of the SURNAME-VARIANT Table associated with the selected cultural perspective the tool shall associate all of its known variants within that cultural perspective, and their variation score (SNV-SCORE) with the query SN segment for use in the evaluation process.
    - If the query SN segment is not included in the subset of the SURNAME-VARIANT Table associated with the selected cultural perspective, and the selected cultural perspective is not "Generic", then the tool shall look up each query SN segment to determine whether it is included in the "Generic" subset of the SURNAME-VARIANT Table.
        - If the query SN segment is included in the "Generic" subset of the SURNAME-VARIANT Table, the tool shall associate all of its known variants within the "Generic" subset, and their variation score (SNV-SCORE) with the query SN segment for use in the evaluation process.
        - If the query SN segment is not included in the "Generic" subset of the SURNAME-VARIANT Table, the tool shall perform no additional Surname Variant processing for this SN segment.

5.1.10.2 Design Notes

1. The type of variation defined for the Anglo contents of the SURNAME-VARIANT Table were determined based on different cultural perspectives using the phonetic workbench to generate variants that would not be handled by a digraph search.
2. Hispanic Surname Variants were gathered from the HNA variant table LAS generated for the State Department.

3. Korean Surname Variants were gathered from the data LAS generated for ORD-C and supplemented with entries in the DNC variant table LAS generated for the State Department.

4. Chinese Surname Variants were gathered from the DNC variant table LAS generated for the State Department.

5.1.10.3 Future Version Notes

1. The developer or user shall be provided mechanisms for adding new variants to the SURNAME-VARIANT Table.

2. The developer or user shall not be allowed to delete variants from the SURNAME-VARIANT Table.

3. The SURNAME-VARIANT Table may support additional cultural perspectives.

4. In the future, the tool may process SN Variants differently based on culture (note that this is part of the justification for separating Generic from Anglo).

6. Evaluate and Score

6.1 Functionality

The tool shall compare each candidate name with the query name to determine whether the candidate name qualifies as a similar name.

In order to determine whether the candidate name is similar to the query name, the tool shall:

- **Evaluate the Surname**;
- **Evaluate the Given Name**;
- **Determine if the SurnameScore exceeds SurnameThreshold**;
- **Determine if the GivenNameScore exceeds GivenNameThreshold;** and
- then **Compute a NameScore & Determine If Potential Match.**

6.1.1 Evaluate Surname

In order to evaluate the Surname, the tool shall:

- First **Determine a SurnameSegmentScore** for each possible pairing of query and candidate SN segments;

- Then **Apply SN Segment Evaluation Factors** to adjust the SurnameSegmentScore (resulting from either a Surname Variant match, Surname Initial match, not exist or unknown match, or a Surname Digraph match) by multiplying the SurnameSegmentScore according to a set of Surname evaluation factors; and

- Finally, **Determine a SurnameScore**.

6.1.1.1 Determine SurnameSegmentScore

The tool shall compare each of the candidate SN segments with the query SN segments to determine a SurnameSegmentScore for each pair of SN segments.

This pairing of SN segments can be represented in an evaluation matrix, such as the one depicted below.

	Granier	Smith
Smyth	SurnameSegmentScore1	SurnameSegmentScore2

SurnameSegmentScore1 = SurnameSegmentScore determined when comparing "Smith" and "Granier".
SurnameSegmentScore2 = SurnameSegmentScore * Evaluation Factor determined when comparing "Smith" and "Smith".

The tool shall determine each SurnameSegmentScore as follows:

- First **Check for Not Exist or Unknown Values (SurnameCheckUnknownNotExist, LastNameUnknownScore, NoLastNameScore)** on the two SN segments.

- If the two SN segments are not a Not Exist or Unknown match, **Check for Surname Variant Match (SurnameCheckVariant, SNV-SCORE)**

- If the two SN segments are not a Not Exist or Unknown match or Surname Variant match, **Check for Surname Initial Match (SurnameCheckInitial, SurnameInitialScore, SurnameExactInitialMatchScore)**

- If the two SN segments are not a Not Exist or Unknown match, or a Surname Variant match, or a Surname Initial match, then the tool shall **Perform a Surname Digraph Comparison (SurnameCheckBias)**

6.1.1.1.1 Check for Not Exist or Unknown Values (SurnameCheckUnknownNotExist, LastNameUnknownScore, NoLastNameScore)

If SurnameCheckUnknownNotExist = "T", then the tool shall determine whether the NoLastNameScore or LastNameUnknownScore can be assigned to the SurnameSegmentScore to handle a SN segment that does not exist or whose value is unknown.

The following table illustrates the SurnameCheckUnknownNotExist conditions and associated values for setting the SurnameSegmentScore:

comparand A	comparand B known	comparand B unknown	comparand B not exist
known	N/A	LastNameUnknownScore	NoLastNameScore
unknown	LastNameUnknownScore	(LastNameUnknownScore + 1)/2	(LastNameUnknownScore +1)/2
not exist	NoLastNameScore	(LastNameUnknownScore +1)/2	(NoLastNameScore + 1)/2

If one comparand is defined as "unknown" and the other comparand is "known", then the tool shall set the SurnameSegmentScore = LastNameUnknownScore.

If one comparand is identified as "unknown", and the other comparand is defined as "not exist", then the tool shall set the SurnameSegmentScore = (LastNameUnknownScore+1)/2.

If one comparand is defined as "known" and the other comparand is "not exist", then the tool shall set the SurnameSegmentScore = NoLastNameScore.

If both comparands are identified as "unknown", then the tool shall set the SurnameSegmentScore = (LastNameUnknownScore+1)/2.

If both comparands are identified as "not exist", then the tool shall set the SurnameSegmentScore = (NoLastNameScore+1)/2.

6.1.1.1.2 Check for Surname Variant Match (SurnameCheckVariant, SNV-SCORE)

If SurnameCheckVariant = "T", the tool shall determine whether a SNV-SCORE can be applied.

For every segment pairing, the tool shall determine whether the two SN segments have been pre-determined to be variants of one another (i.e., defined in the SURNAME-VARIANT Table) by checking to see if the candidate SN segment is present in the list of variants associated with the query SN segment.

If the candidate SN segment is present in the list of variants associated with the query SN segment, then the tool shall set the SurnameSegmentScore = SNV-SCORE associated with the query variant.

6.1.1.1.3 Check for Surname Initial Match (SurnameCheckInitial, SurnameInitialScore, SurnameExactInitialMatchScore)

If SurnameCheckInitial = "T" and the SN segment was not identified as a Surname Variant, then the tool shall determine whether the SurnameInitialScore or SurnameExactInitialMatchScore can be applied.

> If comparand A's SN segment is a single character and comparand B's SN segment is a single character and they match, then the tool shall set the SurnameSegmentScore = SurnameExactInitialMatchScore.
>
> If comparand A's SN segment is a single character and comparand B's SN segment is more than one character and comparand A's SN segment matches the first character of comparand B's SN segment, the tool shall set the SurnameSegmentScore = SurnameInitialScore.

6.1.1.1.4 Perform a Surname Digraph Evaluation

A value from 0.0 to 1.0 shall be calculated based on the number of digraphs which match between two SN segments.

A digraph shall only participate in a match once.

One point shall be awarded for each digraph that participates in a match, thus each digraph match shall result in exactly two points being added to the total digraph score.

The SurnameSegmentScore shall be the total number of points assigned based on the matching digraphs (digraph score), divided by the number of digraphs that occur in the two SN segments.

For example, the SN segments "Garcia" and "Garica" are not an exact match. Of fourteen total digraphs involved in the evaluation, there are four matches, involving 8 digraphs.

```
                    Query SN Segment: Garcia
                  Candidate SN Segment : Garica

                    #G Ga ar rc ci ia a#
                    #G Ga ac cr ri ia a#
                        #G Ga ia a#
```

Therefore, the name receives a digraph score of 8/14 = .57

6.1.1.1.4.1 Apply Surname Left Digraph Bias (SurnameCheckBias)

If SurnameCheckBias = "T", then a bias will be applied so that digraphs on the right end of the strings count less than those on the left in a particular SN segment.

If SurnameCheckBias = "T", the tool shall apply bias by:

- First, assigning the first contributing digraph in each SN segment a weight factor of 1.00, the second contributing digraph a weight factor of .9, and so on until the tenth contributing digraph is reached, at which point, all remaining digraphs shall be assigned a weight factor of .1.

- then determine which digraphs match between the two SN segments;

- sum the weight factors assigned to the matched query SN digraphs with the weight factors assigned to the matched candidate SN digraphs; and

- divide by the sum of all contributing digraphs for both the query SN and the candidate SN.

```
                    SurnameCheckBias : T

                 Query : Moskyovich, FNU
               Candidate :  Markovich, FNU

        -M  Mo  os  sk  ky  yo  ov  vi  ic  ch  h-
      (1.0) (.9) (.8)(.7)(.6) (.5) (.4)(.3)(.2) (.1)(.1)

          -M   Ma  ar  rk   ko   ov  vi  ic  ch  h-
        (1.0) (.9) (.8)(.7)(.6) (.5)(.4)(.3)(.2) (.1)
```

The following digraphs match:

```
                                 -M   ov   vi   ic   ch   h-
Query Weight Factors:         (1.0) (.4)  (.3)  (.2)  (.1)  (.1)
Candidate Weight Factors:     (1.0) (.5)  (.4)  (.3)  (.2)  (.1)
```

$$\frac{\text{Matched Digraphs}}{\text{Total possible Digraphs}} = \frac{(1.0+.4+.3+.2+.1+.1)+(1.0+.5+.4+.3+.2+.1)}{(1.0+.9+.8+.7+.6+.5+.4+.3+.2+.1+.1)+(1.0+.9+.8+.7+.6+.5+.4+.3+.2+.1)}$$

Therefore, the SurnameSegmentScore == (4.6 / 11.1) = 0.41

6.1.1.1.5 Design Notes

1. We may want to support a Right Bias in the future.

6.1.1.2 Apply SN Segment Evaluation Factors

SN Segment Evaluate Factors will be applied, if appropriate, to each segment in the evaluation matrix.

The tool shall determine whether to apply certain SN Segment Evaluation Factors in determining a similar name, based on the application of the following set of logical parameters:

- **Determine Relative Position of SN Segments (SurnameAnchorSegment);**
- **Apply Surname Out of Position Factor (SurnameOutOfPositionFactor);**
- **Apply Surname Anchor Segment Factor (SurnameAnchorSegment, SurnameAnchorFactor)**; and
- **Apply Surname TAQ Factors (SurnameCheckTAQ, SurnameTAQDeleteFactor, SurnameTAQDisregardFactor, SurnameTAQDisregardAbsentFactor, SurnameTAQDeleteAbsentFactor).**

6.1.1.2.1 Determine Relative Position of SN Segments (SurnameAnchorSegment)

In order to determine the relative position in both comparands, the tool shall establish an "index" of a segment based on the SurnameAnchorSegment.

For SurnameAnchorSegment = “none” or “first”, the tool shall count segments from left to right.

For SurnameAnchorSegment = “last”, the tool shall count segments from right to left.

6.1.1.2.2 Apply Surname Out of Position Factor (SurnameOutOfPositionFactor)

If a SN segment is out of position, the SurnameOutOfPositionFactor will always be applied.

If two SN segments are <u>not</u> in the same relative position in both comparands, the tool shall multiply the SurnameSegmentScore by the SurnameOutOfPositionFactor.

In the example cited below, “Smyth” and “Smith” are out of position, and thus the SurnameOutOfPosition factor will be applied to SurnameSegmentScore2.

	Granier	Smith
Smyth	SurnameSegmentScore1	SurnameSegmentScore2

6.1.1.2.3 Apply Surname Anchor Segment Factor (SurnameAnchorFactor, SurnameAnchorSegment)

The SurnameAnchorFactor is used to identify and emphasize the importance of one segment of the surname over another if more than one SN segment exists.

The SurnameAnchorFactor shall never be applied if the SurnameMode is "average" as this would doubly discount the contribution of a single SN segment when determining an overall SN score.

The SurnameAnchorFactor shall never be applied if the SurnameOutOfPositionFactor has already been applied, even if the SurnameAnchorSegment = "first" or "last". Thus, the SN segments must be in position if SurnameAnchorFactor will be applied.

When the SurnameAnchorSegment = "none", neither SN segment will be assigned more weight than the other (i.e., Anchor segment is essentially turned off). When the SurnameAnchorSegment = "first", the first (i.e., left-most) SN segment will be assigned more weight, and when the SurnameAnchorSegment = "last", the last (i.e., right-most) SN segment will be assigned more weight.

If two SN segments are in the same relative position in both comparands (i.e., SurnameOutOfPositionFactor did not apply), and SurnameMode is "lowest" or "highest", and their position is not the SurnameAnchorSegment, and SurnameAnchorSegment = "first" or "last", then the tool shall multiply the SurnameSegmentScore by the SurnameAnchorFactor.

6.1.1.2.4 Apply Surname TAQ Factors (SurnameCheckTAQ, SurnameTAQDeleteFactor, SurnameTAQDisregardFactor, SurnameTAQDisregardAbsentFactor, SurnameTAQDeleteAbsentFactor)

6.1.1.2.4.1 Functionality

DISREGARD TAQs are viewed during evaluation as more important than DELETE TAQs. Two TAQs of the same type (i.e., DELETE or DISREGARD) that do not match are viewed during evaluation as more important than absence of a TAQ type in one comparand and presence of that same TAQ type in the other comparand.

When the SurnameCheckTAQ = "off" or "remove", no TAQ processing shall take place during the evaluation process.

When the SurnameCheckTAQ = "score", TAQs that were identified, removed, and associated with each relevant SN segment during preprocessing shall be factored into the SurnameSegmentScore.

If SurnameCheckTAQ = "score", the tool shall determine which of the following four parameters can be applied to the SurnameSegmentScore:

- **SurnameTAQDisregardFactor;**
- **SurnameTAQDeleteFactor**;
- **SurnameTAQDisregardAbsentFactor**; and
- **SurnameTAQDeleteAbsentFactor**.

TAQ processing shall be performed as follows:

- First, determine whether the query name segment and the candidate name segment each have associated TAQs identified during pre-processing.
- If no TAQs are associated with either segment, then the tool shall not adjust the SurnameSegmentScore (i.e., DELETE TAQs None Occur, DISREGARD TAQs None Occur).
- If one comparand has all DELETE TAQs associated with it and the other comparand has no TAQs associated with it, then the SurnameSegmentScore will be multiplied by the SurnameTAQDeleteAbsentFactor (i.e., DELETE TAQs Absent, DISREGARD TAQs None Occur).
- If one or more DISREGARD TAQs are associated with one comparand and not the other (i.e., either the query name segment or the candidate name segment has one or more DISREGARD TAQs), then the tool shall apply the SurnameTAQDisregardAbsentFactor (i.e., DELETE TAQs >=1 Match, No Match, Absent, None Occur, DISREGARD TAQs Absent).
- If DISREGARD TAQs are present in both comparands, then the tool shall determine whether there are any matches on any of the DISREGARD TAQs:
    - If any matches are found, then the tool shall determine if there are any matches on any DELETE TAQs.
        - If no DELETE TAQs are present, then the tool shall not adjust the SurnameSegmentScore (i.e., DELETE TAQs None Occur, DISREGARD TAQs >=1 Match).
        - If DELETE TAQs are present in both comparands and no matches are found, then the SurnameSegmentScore will be multiplied by the SurnameTAQDeleteFactor (i.e., DELETE TAQs No Match, DISREGARD TAQs >=1 Match).
        - If DELETE TAQs are present in both comparands and a match is found, then the tool shall not adjust the SurnameSegmentScore (i.e., DELETE TAQs >=1 Match, DISREGARD TAQs >=1 Match).
        - If DELETE TAQs are present in one comparand, but not the other, then the SurnameSegmentScore will be multiplied by the SurnameTAQDeleteAbsentFactor (i.e., DELETE TAQs Absent, DISREGARD TAQs >=1 Match).
    - If no match is found, then the SurnameSegmentScore will be multiplied by the SurnameTAQDisregardFactor(i.e., DELETE TAQs >=1 Match, No Match, Absent, None Occur, DISREGARD TAQs No Match).

- If both comparands have all DELETE TAQs associated with them, the tool shall determine if any of the DELETE TAQs match:
    - If there is any match, then the tool shall not adjust the SurnameSegmentScore (i.e., DELETE TAQs >=1 Match DISREGARD TAQs None Occur).
    - If there is no match, then the SurnameSegmentScore will be multiplied by the SurnameTAQDeleteFactor (i.e., DELETE TAQs No Match DISREGARD TAQs None Occur).

The following table describes the conditions governing the application of TAQ parameters as described in the text above:

DELETE TAQ(s)	DISREGARD TAQ(s)	Impact on SurnameSegmentScore
None Occur	None Occur	No Change
None Occur	No Match	Apply SurnameTAQDisregardFactor
None Occur	Absent	Apply SurnameTAQDisregardAbsentFactor
None Occur	>=1 Match	No Change
No Match	None Occur	Apply SurnameTAQDeleteFactor
No Match	No Match	Apply SurnameTAQDisregardFactor
No Match	Absent	Apply SurnameTAQDisregardAbsentFactor
No Match	>=1 Match	Apply SurnameTAQDeleteFactor
Absent	None Occur	Apply SurnameTAQDeleteAbsentFactor
Absent	No Match	Apply SurnameTAQDisregardFactor
Absent	Absent	Apply SurnameTAQDisregardAbsentFactor
Absent	>=1 Match	Apply SurnameTAQDeleteAbsentFactor
>=1 Match	None Occur	No Change
>=1 Match	No Match	Apply SurnameTAQDisregardFactor
>=1 Match	Absent	Apply SurnameTAQDisregardAbsentFactor
>=1 Match	>= 1 Match	No Change

"Match" in the table indicates that the stated TAQ Type occurs in both comparands and at least one of the TAQ Type values occurs in both comparands, i.e., the TAQ Type values are the same. For example, the DELETE TAQ value "Mr" may occur in both comparands.

"No Match" in the table indicates that the stated TAQ Type occurs in both comparands but none of the TAQ Type values occurs in both comparands, i.e., the values are not the same. For example, the single DELETE TAQ value "Mr" may occur in one comparand and the single DELETE TAQ value "Mrs" may occur in the other comparand.

"Absent" in the table indicates that the stated TAQ Type is absent in one of the comparands but occurs in the other comparand. For example, the DELETE TAQ value "Mr" may occur in one comparand and there may be no DELETE TAQ value at all in the other comparand.

"None Occur" in the table indicates that the stated TAQ Type does not occur in either comparand. For example, no DELETE TAQs occur in either comparand.

6.1.1.2.4.2 Future Version Notes

- The SurnameTAQDisregardFactor will be replaced by the highest TAQ-DISREGARD-WEIGHT for each DISREGARD TAQ relationship that is defined in a TAQ-DISREGARD-WEIGHT Table. The TAQ-DISREGARD-WEIGHT Table defines a weighted relationship between two TAQs that occur within a specified cultural boundary or partition. There may be a default TAQ-DISREGARD-WEIGHT established so that only those TAQ relationships that warrant special weighting may be entered into the TAQ-DISREGARD-WEIGHT Table.

- When the TAQ-DISREGARD-WEIGHT Table is implemented, the importance of DISREGARD versus DELETE TAQs may change. For example, MR and MRS are currently defined as DELETE TAQs, and therefore considered less important than other TAQ values. However, their relationship to one another may be treated with more significance in later versions due to gender specification.

6.1.1.3 Determine SurnameScore

In order to determine the SurnameScore, the tool shall perform the following:

- **Compute the Highest SurnameSegmentScore(s) (SurnameMode="Highest")**
- **Compute the Best Combination of SurnameSegmentScore(s) (SurnameMode="Average")**
- **Compute the Lowest SurnameSegmentScore(s) (SurnameMode="Lowest")**

If either comparand has just one SN segment, then the tool shall set the SurnameScore = the Highest (Best) SurnameSegmentScore found in the evaluation matrix.

If more than one segment occurs in both surnames, then the tool shall **Apply the Surname Mode** in its determination of the SurnameScore.

6.1.1.3.1 Compute Highest SurnameSegmentScore(s) (SurnameMode="Highest")

The tool shall compute the highest set of SurnameSegmentScores (includes the highest SurnameSegmentScores) from the evaluation matrix of scores.

During the evaluation of the matrix, a given row or column shall contribute one and only one score.

The tool shall select the combination of matrix values (with no row or column being used more than once) that includes the highest set of SurnameSegmentScores.

In the following example, the highest SurnameSegmentScores will be 1.0 and .57, since 1.0 is the highest SurnameSegmentScore.

	Garcia	Garza
Garica	.57	.62
Garza	.62	1.0

In the following example, the highest SurnameSegmentScores will be 1.0 and .62, since 1.0 is the highest SurnameSegmentScore, and .62 is the next highest SurnameSegmentScore that is in a different row and column combination in the matrix.

	Garcia	Garza	Garza

Garica	.57	.62	.62
Garza	.62	1.0	1.0

6.1.1.3.2 Compute Best Combination of SurnameSegmentScore(s) (SurnameMode="Average")

The tool shall compute the best possible combination of scores (i.e., the Highest Average of SurnameSegmentScores) from the evaluation matrix of scores.

During the evaluation of the matrix, a given row or column shall contribute one and only one score.

The tool shall select the combination of matrix values (with no row or column being used more than once) that gives the highest sum.

In the following example, the best combination of SurnameSegmentScores will be 1.0 and .57, since ((1.0+.57)/2) > ((.62+.62)/2) = .79 > .62.

	Garcia	Garza
Garica	.57	.62
Garza	.62	1.0

6.1.1.3.3 Compute Lowest SurnameSegmentScore(s) (SurnameMode="Lowest")

The tool shall compute the lowest set of SurnameSegmentScores (includes the Lowest SurnameSegmentScores) from the evaluation matrix of scores.

During the evaluation of the matrix, a given row or column shall contribute one and only one score.

The tool shall select the combination of matrix values (with no row or column being used more than once) that includes the lowest set of SurnameSegmentScores.

In the following example, the lowest SurnameSegmentScores will be .57 and 1.0, since .57 is the lowest SurnameSegmentScore.

	Garcia	Garza
Garica	.57	.62
Garza	.62	1.0

In the following example, the lowest SurnameSegmentScores will be .57 and 1.0, since .57 is the lowest SurnameSegmentScore, and 1.0 is the next lowest SurnameSegmentScore that is in a different row and column combination in the matrix.

	Garcia	Garza	Garza
Garica	.57	.62	.62

Garza	.62	1.0	1.0

6.1.1.3.4 Apply Surname Mode (SurnameMode)

If SurnameMode = "highest", the tool shall set the SurnameScore = the highest SurnameSegmentScore found in the evaluation matrix.

If SurnameMode = "average", the tool shall set the SurnameScore = the average of the SurnameSegmentScores found in the evaluation matrix.

If SurnameMode = "lowest", the tool shall set the SurnameScore = the lowest SurnameSegmentScore found in the evaluation matrix.

6.1.1.3.5 Determine SurnameCompressedScore (SurnameCheckCompressed, SurnameCompressedScore)

This function handles names that are essentially the same name but are segmented differently (e.g., "de la Garcia" → "delaGarcia").

If SurnameCheckCompressed = "T", then the tool shall generate a SurnameCompressedScore in the following manner:

- Create query and candidate Compressed SN fields from their original SN fields, by processing segmentation and removal markers, and then eliminating all remaining blanks.
- Compare the query and candidate COMPRESSED SN fields to determine if there is an exact match.
- If there is an exact match of the COMPRESSED SN fields, the tool shall set the SurnameScore to the higher score of the SurnameCompressedScore or the previously calculated SurnameScore.

6.1.2 Evaluate Given Name

In order to evaluate the Given Name, the tool shall:

- First **Determine a GivenNameSegmentScore** for each possible combination of query and candidate GN segments;
- Then, **Apply GN Segment Evaluation Factors** to adjust the GivenNameSegmentScore (resulting from either a Given Name Variant match, Given Name Initial match, not exist or unknown match, or a Given Name Digraph match) by multiplying the GivenNameSegmentScore according to a set of Given Name evaluation factors.

- Finally, **Determine a GivenNameScore**.

6.1.2.1 Determine GivenNameSegmentScore

The tool shall compare each of the candidate GN segments with the query GN segments to determine a GivenNameSegmentScore for each pair of GN segments.

This pairing of GN segments can be represented in an evaluation matrix similar to the one described in the section Determine SurnameSegmentScore.

The tool shall determine each GivenNameSegmentScore as follows:

- First **Check for Not Exist or Unknown Values (GivenNameCheckUnknownNotExist, FirstNameUnknownScore, NoFirstNameScore)** on the two GN segments.
- If the two GN segments are not a Not Exist or Unknown match, **Check for Given Name Variant Match (GivenNameCheckVariant, GNV-SCORE)**
- If the two GN segments are not a Not Exist or Unknown match or Given Name Variant match, **Check for Given Name Initial Match (GivenNameCheckInitial, GivenNameInitialScore, GivenNameExactInitialMatchScore)**
- If the two GN segments are not a Not Exist or Unknown match, or a Given Name Variant match, or a Given Name Initial match, then the tool shall **Perform a Given Name Digraph Comparison (GivenNameCheckBias)**

6.1.2.1.1 Check for Not Exist or Unknown Values (GivenNameCheckUnknownNotExist, FirstNameUnknownScore, NoFirstNameScore)

If GivenNameCheckUnknownNotExist = "T", then the tool shall determine whether the NoFirstNameScore or FirstNameUnknownScore can be assigned to the GivenNameSegmentScore to handle a GN segment that does not exist or whose value is unknown.

The following table illustrates the GivenNameCheckUnknownNotExist conditions and associated values for setting the GivenNameSegmentScore:

comparand A	**comparand B known**	**comparand B unknown**	**comparand B not exist**
known	N/A	FirstNameUnknownScore	NoFirstNameScore
unknown	FIrstNameUnknownScore	(FirstNameUnknownScore + 1)/2	(FirstNameUnknownScore +1)/2
notexist	NoFirstNameScore	(FirstNameUnknownScore +1)/2	(NoFirstNameScore + 1)/2

If one comparand is defined as "unknown" and the other comparand is "known", then the tool shall set the GivenNameSegmentScore = FirstNameUnknownScore.

If one comparand is defined as "known" and the other comparand is "not exist", then the tool shall set the GivenNameSegmentScore = NoFirstNameScore.

If both comparands are identified as "not exist" or both are identified as "unknown", then the tool shall set the GivenNameSegmentScore = (FirstNameUnknownScore + 1)/2.

If both comparands are identified as either "not exist" or "unknown", and the comparands are not defined the same, then the tool shall set the GivenNameSegmentScore = (FirstNameUnknownScore+1)/2.

If both comparands are identified as "unknown", then the tool shall set the GivenNameSegmentScore = (FirstNameUnknownScore+1)/2.

If both comparands are identified as "not exist", then the tool shall set the GivenNameSegmentScore = (NoFirstNameScore+1)/2.

6.1.2.1.2 Check for Given Name Variant Match (GivenNameCheckVariant, GNV-SCORE)

If GivenNameCheckVariant = "T", the tool shall determine whether the GNV-SCORE can be applied.

For every segment pairing, the tool shall determine whether the two GN segments have been pre-determined to be variants of one another (i.e., defined in the GIVEN-NAME-VARIANT Table) by checking to see if the candidate GN segment is present in the list of variants associated with the query GN segment.

If the candidate GN segment is present in the list of variants associated with the query GN segment, then the tool shall set the GivenNameSegmentScore = GNV-SCORE associated with the query variant.

6.1.2.1.3 Check for Given Name Initial Match (GivenNameCheckInitial, GivenNameInitialScore, GivenNameExactInitialMatchScore)

If GivenNameCheckInitial = "T" and the GN segment was not identified as a Given Name Variant, then the tool shall determine whether the GivenNameInitialScore or GivenNameExactInitialMatchScore can be applied.

> If comparand A's GN segment is a single character and comparand B's GN segment is a single character and they match, then the tool shall set the GivenNameSegmentScore = GivenNameExactInitialMatchScore.
>
> If comparand A's GN segment is a single character and comparand B's GN segment is more than one character and comparand A's GN segment matches the first character of comparand B's GN segment, the tool shall set the GivenNameSegmentScore = GivenNameInitialScore.

6.1.2.1.4 Perform Given Name Digraph Evaluation

A value from 0.0 to 1.0 shall be calculated based on the number of digraphs which match between two Given Name segments.

A digraph shall only participate in a match once.

One point shall be awarded for each digraph that participates in a match, thus each digraph match shall result in exactly two points being added to the total digraph score.

The GivenNameSegmentScore shall be the total number of points assigned based on the matching digraphs (digraph score), divided by the number of digraphs that occur in the two GN segments.

6.1.2.1.4.1 Apply Given Name Left Digraph Bias (GivenNameCheckBias)

If GivenNameCheckBias = "T", then a bias will be applied so that digraphs on the right end of the strings count less than those on the left in a particular GN segment.

If GivenNameCheckBias = "T", the tool shall apply bias by:

- First, assigning the first contributing digraph in each GN segment a weight factor of 1.00, the second contributing digraph a weight factor of .9, and so on until the tenth contributing digraph is reached, at which point, all remaining digraphs shall be assigned a weight factor of .1.
- then determine which digraphs match between the two GN segments;
- sum the weight factors assigned to the matched query GN digraphs with the weight factors assigned to the matched candidate GN digraphs; and
- divide by the sum of all contributing digraphs for both the query GN and the candidate GN.

6.1.2.1.5 Design Notes

1. We may want to support a Right Bias in the future.

6.1.2.2 Apply GN Segment Evaluation Factors

The tool shall determine whether to apply certain GN Segment Evaluation Factors in determining a similar name, based on the application of the following set of logical parameters:

- **Determine Relative Position of GN Segments (GivenNameAnchorSegment);**
- **Apply Given Name Out of Position Factor (GivenNameOutOfPositionFactor);**

- **Apply Given Name Anchor Segment Factor (GivenNameAnchorSegment, GivenNameAnchorFactor);** and
- **Apply Given Name TAQ Factors (GivenNameCheckTAQ, GivenNameTAQDeleteFactor, GivenNameTAQDisregardFactor, GivenNameTAQDisregardAbsentFactor, GivenNameTAQDeleteAbsentFactor).**

6.1.2.2.1 Determine Relative Position of GN Segments (GivenNameAnchorSegment)

In order to determine the relative position in both comparands, the tool shall establish an "index" of a segment based on the GivenNameAnchorSegment.

For GivenNameAnchorSegment = “none” or “first”, the tool shall count segments from left to right.

For GivenNameAnchorSegment = “last”, the tool shall count segments from right to left.

6.1.2.2.2 Apply Given Name Out of Position Factor (GivenNameOutOfPositionFactor)

If a GN segment is out of position, the GivenNameOutOfPositionFactor will always be applied.

If two GN segments are <u>not</u> in the same relative position in both comparands, the tool shall multiply the GivenNameSegmentScore by the GivenNameOutOfPositionFactor.

In the example cited below, “Jeffrey” and “Jeffrey” are out of position, and thus the SurnameOutOfPosition factor will be applied to SurnameSegmentScore3.

	Jeffrey	Andrew
A	SurnameSegmentScore1	SurnameSegmentScore2
Jeffrey	SurnameSegmentScore3	SurnameSegmentScore4

6.1.2.2.3 Apply Given Name Anchor Segment Factor (GivenNameAnchorFactor, GivenNameAnchorSegment)

The GivenNameAnchorFactor is used to identify and emphasize the importance of one segment of the given name over another if more than one GN segment exists.

The GivenNameAnchorFactor shall never be applied if the GivenNameMode is “average” as this would doubly discount the contribution of a single GN segment when determining an overall GN score.

The GivenNameAnchorFactor shall never be applied if the GivenNameOutOfPositionFactor has already been applied, even if the GivenNameAnchorSegment = “first” or “last”. Thus, the GN segments must be in position if GivenNameAnchorFactor will be applied.

When the GivenNameAnchorSegment = "none", neither GN segment will be assigned more weight than the other (i.e., Anchor segment is essentially turned off). When the GivenNameAnchorSegment = "first", the first (i.e., left-most) GN segment will be assigned more weight, and when the GivenNameAnchorSegment = "last", the last (i.e., right-most) GN segment will be assigned more weight.

If two GN segments are in the same relative position in both comparands (i.e., GivenNameOutOfPositionFactor did not apply), and their position is not the GivenNameAnchorSegment, and GivenNameAnchorSegment = "first" or "last", and GivenNameMode is "lowest" or "highest", then the tool shall multiply the GivenNameSegmentScore by the GivenNameAnchorFactor.

6.1.2.2.4 Apply Given Name TAQ Factors (GivenNameCheckTAQ, GivenNameTAQDeleteFactor, GivenNameTAQDisregardFactor, GivenNameTAQDisregardAbsentFactor, GivenNameTAQDeleteAbsentFactor)

6.1.2.2.4.1 Functionality

DISREGARD TAQs are viewed as more important than DELETE TAQs. Two TAQs of the same type (i.e., DELETE or DISREGARD) that do not match are viewed as more important than absence of a TAQ type in one comparand and presence of that same TAQ type in the other comparand.

When the GivenNameCheckTAQ = "off" or "remove", no TAQ processing will take place during the evaluation process.

When the GivenNameCheckTAQ = "score", TAQs that were identified, removed, and associated with each relevant GN segment during preprocessing will be factored into the GivenNameSegmentScore.

If GivenNameCheckTAQ = "score", the tool shall determine which of the following four parameters can be applied to the GIvenNameSegmentScore:

- **GivenNameTAQDisregardFactor;**
- **GivenNameTAQDeleteFactor**;
- **GivenNameTAQDisregardAbsentFactor**; and
- **GivenNameTAQDeleteAbsentFactor**.

TAQ processing shall be performed as follows:

- First, determine whether the query name segment and the candidate name segment each have associated TAQs identified during pre-processing.

- If no TAQs are associated with either segment, then the tool shall not adjust the GIvenNameSegmentScore (i.e., DELETE TAQs None Occur, DISREGARD TAQs None Occur).

- If one comparand has all DELETE TAQs associated with it and the other comparand has no TAQs associated with it, then the GivenNameSegmentScore will be multiplied by the GivenNameTAQDeleteAbsentFactor (i.e., DELETE TAQs Absent, DISREGARD TAQs None Occur).

- If one or more DISREGARD TAQs are associated with one comparand and not the other (i.e., either the query name segment or the candidate name segment has one or more DISREGARD TAQs), then the tool shall apply the GivenNameTAQDisregardAbsentFactor (i.e., DELETE TAQs >=1 Match, No Match, Absent, None Occur, DISREGARD TAQs Absent).

- If DISREGARD TAQs are present in both comparands, then the tool shall determine whether there are any matches on any of the DISREGARD TAQs:

    - If any matches are found, then the tool shall determine if there are any matches on any DELETE TAQs.

        - If no DELETE TAQs are present, then the tool shall not adjust the GivenNameSegmentScore (i.e., DELETE TAQs None Occur, DISREGARD TAQs >=1 Match).

        - If DELETE TAQs are present in both comparands and no matches are found, then the GivenNameSegmentScore will be multiplied by the GivenNameTAQDeleteFactor (i.e., DELETE TAQs No Match, DISREGARD TAQs >=1 Match).

        - If DELETE TAQs are present in both comparands and a match is found, then the tool shall not adjust the GivenNameSegmentScore (i.e., DELETE TAQs >=1 Match, DISREGARD TAQs >=1 Match).

        - If DELETE TAQs are present in one comparand, but not the other, then the GivenNameSegmentScore will be multiplied by the GivenNameTAQDeleteAbsentFactor (i.e., DELETE TAQs Absent, DISREGARD TAQs >=1 Match).

- If no match is found, then the GivenNameSegmentScore will be multiplied by the GivenNameTAQDisregardFactor (i.e., DELETE TAQs >=1 Match, No Match, Absent, None Occur, DISREGARD TAQs No Match).

- If both comparands have all DELETE TAQs associated with them, the tool shall determine if any of the DELETE TAQs match:

- If there is any match, then the tool shall not adjust the GivenNameSegmentScore (i.e., DELETE TAQs >=1 Match, DISREGARD TAQs None Occur).

- If there is no match, then the GIvenNameSegmentScore will be multiplied by the GivenNameTAQDeleteFactor (i.e., DELETE TAQs No Match, DISREGARD TAQs None Occur).

The following table describes the conditions governing the application of TAQ as described in the text above:

DELETE TAQ(s)	DISREGARD TAQ(s)	Impact on GivenNameSegmentScore
None Occur	None Occur	No Change
None Occur	No Match	Apply GivenNameTAQDisregardFactor
None Occur	Absent	Apply GivenNameTAQDisregardAbsentFactor
None Occur	>=1 Match	No Change
No Match	None Occur	Apply GivenNameTAQDeleteFactor
No Match	No Match	Apply GivenNameTAQDisregardFactor
No Match	Absent	Apply GivenNameTAQDisregardAbsentFactor
No Match	>=1 Match	Apply GivenNameTAQDeleteFactor
Absent	None Occur	Apply GivenNameTAQDeleteAbsentFactor
Absent	No Match	Apply GivenNameTAQDisregardFactor
Absent	Absent	Apply GivenNameTAQDisregardAbsentFactor
Absent	>=1 Match	Apply GivenNameTAQDeleteAbsentFactor
>=1 Match	None Occur	No Change
>=1 Match	No Match	Apply GivenNameTAQDisregardFactor
>=1 Match	Absent	Apply GivenNameTAQDisregardAbsentFactor
>=1 Match	>= 1 Match	No Change

"Match" in the table indicates that the stated TAQ Type occurs in both comparands and at least one of the TAQ Type values occurs in both comparands, i.e., the TAQ Type values are the same. For example, the DELETE TAQ value "Mr" may occur in both comparands.

"No Match" in the table indicates that the stated TAQ Type occurs in both comparands but none of the TAQ Type values occurs in both comparands, i.e., the values are not the same. For example, the single DELETE TAQ value "Mr" may occur in one comparand and the single DELETE TAQ value "Mrs" may occur in the other comparand.

"Absent" in the table indicates that the stated TAQ Type is absent in one of the comparands but occurs in the other comparand. For example, the DELETE TAQ value "Mr" may occur in one comparand and there may be no DELETE TAQ value at all in the other comparand.

"None Occur" in the table indicates that the stated TAQ Type does not occur in either comparand. For example, no DELETE TAQs occur in either comparand.

6.1.2.2.4.2 Future Version Notes

- The GivenNameTAQDisregardFactor will be replaced by the highest TAQ-DISREGARD-WEIGHT for each DISREGARD TAQ relationship that is defined in a TAQ-DISREGARD-WEIGHT Table. The TAQ-DISREGARD-WEIGHT Table defines a weighted relationship between two TAQs that occur within a specified cultural boundary or partition. There may be a default TAQ-DISREGARD-WEIGHT established so that only those TAQ relationships that warrant special weighting may be entered into the TAQ-DISREGARD-WEIGHT Table.

6.1.2.3 Determine GivenNameScore

In order to determine the GivenNameScore, the tool shall:

- **Compute the Highest GivenNameSegmentScore(s) (SurnameMode="Highest");**
- **Compute the Best Combination of GivenNameSegmentScore(s) (SurnameMode="Average"); and**
- **Compute the Lowest GivenNameSegmentScore(s) (SurnameMode="Lowest");**

If either comparand has just one GN segment, then the tool shall set the GivenNameScore = the Highest (Best) GivenNameSegmentScore found in the evaluation matrix.

If more than one segment occurs in both Given Names, then the tool shall **Apply the Given Name Mode** in its determination of the GivenNameScore.

6.1.2.3.1 Compute Highest GivenNameSegmentScore(s) (GivenNameMode="Highest")

The tool shall compute the highest set of GivenNameSegmentScores (includes the highest GivenNameSegmentScores) from the evaluation matrix of scores.

During the evaluation of the matrix, a given row or column shall contribute one and only one score.

The tool shall select the combination of matrix values (with no row or column being used more than once) that includes the highest set of GivenNameSegmentScores.

In the following example, the highest GivenNameSegmentScores will be 1.0 and .57, since 1.0 is the highest GivenNameSegmentScore.

	Garcia	Garza
Garica	.57	.62
Garza	.62	1.0

In the following example, the highest GivenNameSegmentScores will be 1.0 and .62, since 1.0 is the highest GivenNameSegmentScore, and .62 is the next highest GivenNameSegmentScore that is in a different row and column combination in the matrix.

	Garcia	Garza	Garza
Garica	.57	.62	.62
Garza	.62	1.0	1.0

6.1.2.3.2 Compute Best Combination of GivenNameSegmentScore(s) (GivenNameMode="Average")

The tool shall compute the best possible combination of scores (i.e., the Highest Average of GivenNameSegmentScores) from the evaluation matrix of scores.

During the evaluation of the matrix, a given row or column shall contribute one and only one score.

The tool shall select the combination of matrix values (with no row or column being used more than once) that gives the highest sum.

In the following example, the best combination of GivenNameSegmentScores will be 1.0 and .57, since ((1.0+.57)/2) > ((.62+.62)/2) = .79 > .62.

	Garcia	Garza
Garica	.57	.62
Garza	.62	1.0

6.1.2.3.3 Compute Lowest GivenNameSegmentScore(s) (GivenNameMode="Lowest")

The tool shall compute the lowest set of GivenNameSegmentScores (includes the Lowest GivenNameSegmentScores) from the evaluation matrix of scores.

During the evaluation of the matrix, a given row or column shall contribute one and only one score.

The tool shall select the combination of matrix values (with no row or column being used more than once) that includes the lowest set of GivenNameSegmentScores.

In the following example, the lowest GivenNameSegmentScores will be .57 and 1.0, since .57 is the lowest GivenNameSegmentScore.

	Garcia	Garza
Garica	.57	.62
Garza	.62	1.0

In the following example, the lowest GivenNameSegmentScores will be .57 and 1.0, since .57 is the lowest GivenNameSegmentScore, and 1.0 is the next lowest GivenNameSegmentScore that is in a different row and column combination in the matrix.

	Garcia	Garza	Garza
Garica	.57	.62	.62
Garza	.62	1.0	1.0

6.1.2.3.4 Apply Given Name Mode (GivenNameMode)

If GivenNameMode = "highest", the tool shall set the GivenNameScore = the highest GivenNameSegmentScore found in the evaluation matrix.

If GivenNameMode = "average", the tool shall set the GivenNameScore = the average of the GivenNameSegmentScores found in the evaluation matrix.

If GivenNameMode = "lowest", the tool shall set the GivenNameScore = the lowest GivenNameSegmentScore found in the evaluation matrix.

6.1.2.3.5 Determine GivenNameCompressedScore (GivenNameCheckCompressed, GivenNameCompressedScore)

This function handles names that are essentially the same name but are segmented differently (e.g., "Anne Marie" → "AnneMarie").

If GivenNameCheckCompressed = "T", then the tool shall generate a GivenNameCompressedScore in the following manner:

- Create query and candidate Compressed GN fields from their original GN fields, by processing segmentation and removal markers, and then eliminating all remaining blanks.
- Compare the query and candidate COMPRESSED GN fields to determine if there is an exact match.
- If there is an exact match of the COMPRESSED GN fields, the tool shall set the GivenNameScore to the higher score of the GivenNameCompressedScore or the previously calculated GivenNameScore.

6.1.3 Determine if SurnameScore exceeds SurnameThreshold

The tool shall determine whether the candidate name is a potential match by checking to see if the SurnameScore exceeds the SurnameThreshold prior to returning the candidate name as a potential match.

If the SurnameScore exceeds the SurnameThreshold, then the tool shall determine that the candidate name is still a potential match.

If the SurnameScore does not exceed the SurnameThreshold, then the tool shall determine that the candidate name is no longer a potential match.

Even if the candidate name is no longer considered a potential match, the tool shall continue processing in the event that all evaluated names were requested to be scored and returned marked as match or no match.

6.1.4 Determine if GiveNameScore exceeds GivenNameThreshold

The tool shall determine whether the candidate name is a potential match by checking to see if the GivenNameScore exceeds the GivenNameThreshold prior to returning the candidate name as a potential match.

If the GivenNameScore exceeds the GivenNameThreshold, then the tool shall determine that the candidate name is still a potential match.

If the GivenNameScore does not exceed the GivenNameThreshold, then the tool shall determine that the candidate name is no longer a potential match.

Even if the candidate name is no longer considered a potential match, the tool shall continue processing in the event that all evaluated names were requested to be scored and returned marked as match or no match.

6.1.5 Compute NameScore & Determine If Potential Match (NameThreshold, SurnameWeight, GivenNameWeight)

If the SurnameWeight = GivenNameWeight = 0, then the tool shall set NameScore = 0.

If the SurnameWeight = GivenNameWeight <> 0, then the tool shall assign NameScore = (SurnameScore + GivenNameScore)/2.

If the SurnameWeight <> GivenNameWeight then the tool shall assign

$$\text{NameScore} = \frac{(\text{SurnameScore}*\text{SurnameWeight}) + (\text{GivenNameScore}*\text{GivenNameWeight})}{(\text{SurnameWeight} + \text{GivenNameWeight})}$$

The tool shall then determine whether the candidate name is a potential match by checking to see if the NameScore exceeds the NameThreshold prior to returning the candidate name as a potential match.

If the NameScore exceeds the NameThreshold, then the tool shall determine that the candidate name is still a potential match.

Developers will be able to establish their own method to determine if a candidate name is a potential match. This will enable developers to integrate other data elements and other criteria in the final name score, if desired. Note that developers may or may not choose to utilize the SurnameWeight and GivenNameWeight factors in their method.

The tool shall populate the Results List with a candidate name based on whether it is identified as a potential match. If no Results List is being constructed, then the tool shall return the candidate name and its associated scores.

6.2 Design Notes

1. We considered performing an exact match on the name prior to performing the "fuzzy" matching, but decided that the overhead of checking every candidate name as an exact match was more than the cost of performing the "fuzzy" match when there is an exact match – our assumption is that the tool will be evaluating more similar names than exact match names.

2. The following parameters have been supported by earlier versions of DNC and are not proposed to be included in the tool.

   - The following parameters were used with the DP2-based pass-1 search for DNC, and were supplanted by the COF processor and, therefore, are now no longer functional in DNC:

     - **KICKOUT**
     - **PARTITION**
     - **TEST VALUE**
     - **PROXRETURN**

   - The following parameters were specifically supported in DNC for the State Department:

     - **REFULEVx (REFULEV0 - REFULEV4)**
     - **DOBFACTOR**
     - **FIXLASTSEG**
     - **CHKSUBSTR – (DNC does not use except for 1 COB)**
     - **SUBSCORE – (DNC does not use)**
     - **MINSEGLEN – (DNC does not use)**
     - **LTRIGRAPH - (related to COF processing)**
     - **NTRIGRAPH - (related to COF processing)**

7. Produce and Manage Results

7.1 Functionality

7.1.1 Define Criteria for Results List

7.1.1.1 Functionality

The Results list is defined as either an unordered candidate name list, or an ordered candidate name list ordered by the relative probability that each candidate name is similar to a specified query name. This probability is represented by the final NameScore. The Results List may include both similar and dissimilar names, depending on the criteria for defining the results.

When a set of candidate names is to be evaluated, the tool shall enable the developer to define the criteria for producing and managing their own Results List. These criteria shall include:

- establishing a type of Results List :
    - 1 = an unordered list of all candidate names whose name score exceeds a pre-defined name threshold (e.g., if the threshold = 0, all candidate names will be returned in an unordered list);
    - 2 = an ordered list of all candidate names whose name score exceeds a pre-defined name threshold (e.g., if the threshold = 0, all candidate names will be returned in an ordered list);
    - 3 = an ordered list of the top X candidate names whose name score exceeds a pre-defined name threshold, and where X is a number;
- establishing a size limit for the Results list, which in effect defines the value of X for producing the top X candidate names;
- defining a SurnameThreshold;
- defining a GivenNameThreshold; and
- defining a NameThreshold.

If the NameThreshold is set to 0, the tool shall return all candidate names in a Results List, unless a Results List size limit is established.

7.1.1.2 Design Notes

1. The developer can define the top X candidate names by establishing their own Results list and attaching it to the query name. In establishing their own Results list, the developer is in effect specifying its size, i.e., the value of X.

7.1.2 Produce Results

Only those candidate names whose NameScore is greater than the NameThreshold will be included in the Result List.

The tool shall produce the Result List by sorting the candidate names in descending order by their NameScore.

If two candidate NameScores are the same, then the tool shall order the same scored candidate data according to the following rules:

- first order the candidate names in descending order by each candidate's SurnameScore.
- if two candidate's SurnameScores are the same, then do the following:
    - if SurnameMode = "average" or "lowest", then do the following:
        - first order the candidate names in descending order by each candidate's GivenNameScore.
        - if two candidate's GivenNameScores are the same, then do the following:
            - If GivenNameMode = "average" or "lowest", then do the following:
                - first order the candidate names in ascending order by the difference in the number of SN segments between the candidate name and the query name.
                - if the difference in the number of SN segments between the candidate name and the query name is the same, then order the candidate names in ascending order by the difference in the number of GN segments between the candidate name and the query name.
            - If GivenNameMode = "highest", then do the following:
                - first order the candidate names in descending order by each candidate's next highest GivenNameSegmentScore.
                - if two candidate's next highest GivenNameSegmentScores are still all the same, then continue to evaluate the next highest GivenNameSegmentScores (up to n, where n is the greater

number of Given Name Segments in the two evaluation Given Names), and order the candidate names in descending order by each candidate's next highest GivenNameSegmentScore. If one of the evaluation Given Names has fewer segments than the other evaluation Given Name, its missing GivenNameSegmentScores will be set to .50 in order to evaluate them.

- if two candidate's GivenNameSegmentScores are all the same, then order the candidate names in ascending order by the difference in the number of SN segments between the candidate name and the query name.
- if the difference in the number of SN segments between the candidate name and the query name is the same, then order the candidate names in ascending order by the difference in the number of GN segments between the candidate name and the query name.

- If SurnameMode = "highest", then do the following:
    - first order the candidate names in descending order by each candidate's next highest SurnameSegmentScore.
    - if two candidate's next highest SurnameSegmentScores are still all the same, then continue to evaluate the next highest SurnameSegmentScores (up to n, where n is the greater number of Surname Segments in the two evaluation surnames), and order the candidate names in descending order by each candidate's next highest SurnameSegmentScore. If one of the evaluation surnames has fewer segments than the other evaluation surname, its missing SurnameSegmentScores will be set to .50 in order to evaluate them.
        - if two candidate's next highest SurnameSegmentScores are all the same, then do the following:
            - first order the candidate names in descending order by each candidate's GivenNameScore.
            - if two candidate's GivenNameScores are the same, then do the following:
                - if GivenNameMode = "average" or "lowest", then do the following:

    - first order the candidate names in ascending order by the difference in the number of SN segments between the candidate name and the query name.

    - if the difference in the number of SN segments between the candidate name and the query name is the same, then order the candidate names in ascending order by the difference in the number of GN segments between the candidate name and the query name.

- if GivenNameMode = "highest", then do the following:

    - first order the candidate names in descending order by each candidate's next highest GivenNameSegmentScore.

    - if two candidate's next highest GivenNameSegmentScores are still all the same, then continue to evaluate the next highest GivenNameSegmentScores (up to n, where n is the greater number of Given Name Segments in the two evaluation Given Names), and order the candidate names in descending order by each candidate's next highest GivenNameSegmentScore. If one of the evaluation Given Names has fewer segments than the other evaluation Given Name, its missing GivenNameSegmentScores will be set to .50 in order to evaluate them.

        - if two candidate's GivenNameSegmentScores are all the same, then order the candidate names in ascending order by the difference in the number of SN segments between the candidate name and the query name.

        - if the difference in the number of SN segments between the candidate name and the query name is the same, then order the candidate names in ascending order by the difference in the number of GN segments between the candidate name and the query name.

If the developer specified the top X option, then the tool shall produce a Result List that contains the X candidate names that are determined after sorting, to be the most likely matches.

The tool shall provide the developer the capability to establish custom results ordering methods, if desired.

7.1.3 Retrieve Results

The tool shall provide the developer the capability to retrieve matched candidate names from the Results List and retrieve additional information about the candidate names to include at a minimum, the Given Name field, the Surname field, the GivenNameScore, the SurnameScore, and other data that the developer may have defined.

8. EVALUATION FACTORS and PARAMETERS

8.1 SurnameCheckInitial (previously known as ISSNINITL)

The SurnameCheckInitial indicates whether a single character in the surname segment will be treated as an initial. When SurnameCheckInitial = "T", single characters in the query SN segment are treated as initials and, if they match on a candidate SN segment, the tool will usually set the SurnameSegmentScore = SurnameInitialScore (except when there is an exact match on initials, in which case the SurnameSegmentScore = (1+SurnameInitialScore)/2 : Exact matches on initials are not considered "exact matches" because the initial may represent two different name segments).

Initials are relatively uncommon in the surname field. In some cases, such as Chinese names, single characters are common in the surname field, in which case one will not want a single letter to be treated as an initial, but rather, treated as a name. If treated as a name, a single character will be analyzed using digraph matching, and will generally be given very little value, unless it matches on an identical one character name. In such cases, SurnameCheckInitial should be set to "F".

SurnameCheckInitial
Possible Settings: {T,F}
Default: {F}

8.2 SurnameCheckVariant (previously known as CHKVARIANT)

In many cases there are variant spellings for a surname that do not share many common digraphs. One possible solution to this problem is the use of the SurnameCheckVariant parameter. When SurnameCheckVariant = "T", a table containing Surname Variants is referenced during the evaluation as well.

SurnameCheckVariant
Possible Settings: {T, F}
Default: {F}

8.3 SurnameCheckBias (previously known as LDIBIAS)

The SurnameCheckBias was designed to aid in the analysis of Russian names and other naming systems which make use of complex morphological endings. If SurnameCheckBias = "T", the tool places more emphasis upon the beginning digraphs of a particular name and de-emphasizes later digraphs. This is done to eliminate the effect of the morphological endings common to Russian names. Because the names are generally long, and many of the names end with the same endings (such as -ovich), candidates that were not very good were being easily returned because there were many digraph matches. When SurnameCheckBias = "T", and when calculating a SurnameSegmentScore, the first digraph in the both the query and candidate SN segments is assigned a weight factor of 1.00, the second digraph is assigned a weight factor of.9, and so forth until .1 is reached, at which point, the remainder of the digraphs are assigned a weight factor of .1.

If GivenNameCheckBias = "T", the tool shall apply bias by:

- First, assigning the first contributing digraph in each GN segment a weight factor of 1.00, the second contributing digraph a weight factor of .9, and so on until the tenth contributing digraph is reached, at which point, all remaining digraphs shall be assigned a weight factor of .1.
- then determine which digraphs match between the two GN segments;
- sum the weight factors assigned to the matched query GN digraphs with the weight factors assigned to the matched candidate GN digraphs; and
- divide by the sum of all contributing digraphs for both the query GN and the candidate GN.

```
                 GivenNameCheckBias :  T

                 Query : Moskyovich, FNU
               Candidate :  Markovich, FNU

         -M  Mo  os  sk  ky  yo  ov  vi  ic  ch  h-
       (1.0) (.9) (.8)(.7)(.6) (.5) (.4)(.3)(.2) (.1)(.1)

          -M    Ma  ar  rk   ko   ov  vi  ic  ch  h-
        (1.0) (.9) (.8)(.7)(.6) (.5)(.4)(.3)(.2) (.1)
```

The following digraphs match:

```
                              -M   ov    vi   ic   ch   h-
Query Weight Factors:       (1.0) (.4)  (.3)  (.2)  (.1)  (.1)
Candidate Weight Factors:   (1.0) (.5)  (.4)  (.3)  (.2)  (.1)
```

$$\frac{\text{Matched Digraphs}}{\text{Total possible Digraphs}} = \frac{(1.0+.4+.3+.2+.1+.1)+(1.0+.5+.4+.3+.2+.1)}{(1.0+.9+.8+.7+.6+.5+.4+.3+.2+.1+.1)+(1.0+.9+.8+.7+.6+.5+.4+.3+.2+.1)}$$

Therefore, the GivenNameSegmentScore == (4.6 / 11.1) = 0.41

There are some problems that may occur when SurnameCheckBias = "T." For example, SurnameCheckBias was set to "T" during the LQA for Poland because of the high frequency morphological endings. However, as a result, a common surname root such as "Kowal" returned names such as "Kowalczyk," "Kowalewska," "Kowalewski," "Kowalska, "Kowalik," "Kowalow," "Kowal," and "Kowalkowska."(Memo# L94289) Therefore, names which were often not good hits were returned because of the additional weight placed upon the beginning digraphs in a name. Please see memos L94289 and L94290 for further explanation of the possible problems associated with SurnameCheckBias.

When the SurnameCheckBias = "T", more emphasis is placed upon the beginning digraphs of a name. When the SurnameCheckBias = "F", equal value is placed upon all matching digraphs.

SurnameCheckBias
Possible Settings: {T,F}
Default: {F}

8.4 SurnameCheckUnknownNotExist, LastNameUnknownScore, NoLastNameScore

Surname Check Non-existent value – used to assign a higher score to a SN segment if there is no value in one comparand SN segment, yet there is a value in the other comparand.

Query : Malcolm LNU
Candidate : Malcolm Shabaz

If SurnameCheckUnknownNotExist = "F", the SurnameSegmentScore for "LNU" compared to "Shabaz" would be 0. With SurnameCheckUnknownNotExist = "T", the SurnameSegmentScore for "LNU" compared to "Shabaz" will be set to the LastNameUnknownScore. The LastNameUnknownScore will be set fairly high to accommodate for missing values when it is unclear whether there should or should not be a value. This would result in a higher SurnameSegmentScore which means that the candidate will be more likely to appear in the TOP X results as well as exceed the NameThreshold if it is set above 0.

SurnameCheckUnknownNotExist : T

Query : Malcolm NLN

Candidate : Malcolm Shabaz

In the second example above, the SurnameSegmentScore for "NLN" compared to "Shabaz" will be set to the NoLastNameScore. The NoLastNameScore will typically be very low to accommodate for the fact that there is no last name defined in the query but a last name appears in the candidate. Thus, the candidate is not a likely match.

SurnameCheckUnknownNotExist
Possible Settings: {T, F}
Default: {F}

NoLastNameScore
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.80}

LastNameUnknownScore
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.85}

8.5 SurnameCheckCompressed, SurnameCompressedScore

SurnameCheckCompressed
Possible Settings: {T,F}
Default: {F}

SurnameCompressedScore
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.9}

8.6 SurnameAnchorSegment, SurnameAnchorFactor (previously known as ANCHSEG, ANCHVAL)

In order to determine the relative position in both comparands, the tool shall establish an "index" of a segment based on the SurnameAnchorSegment. For SurnameAnchorSegment = "none" or "first", the tool shall count segments from left to right. For SurnameAnchorSegment = "last", the tool shall count segments from right to left.

Either SurnameOutOfPositionFactor or SurnameAnchorFactor, but not both, can be applied to the SurnameSegmentScore. Thus, if SurnameOutOfPositionFactor has already been applied, then the SurnameAnchorFactor can not be applied, even if the SurnameAnchorSegment = "first" or "last". If SurnameOutOfPositionFactor does not apply, then the SurnameAnchorFactor may apply if SurnameMode is "highest" or "lowest". If SurnameMode is "average", the SurnameAnchorFactor is not applied.

The SurnameAnchorSegment is used in compound surnames to emphasize the importance of one segment of the name over another. For example, when dealing with Portuguese names, it is the second (i.e., last) surname which is more important, whereas when dealing with other Hispanic names it is generally the first surname which is of primary importance.

When the SurnameAnchorSegment = "none", neither segment in the name is given more weight (i.e., basically the SurnameAnchorSegment is turned off). When the SurnameAnchorSegment = "first", both comparand segments are left-aligned and the left-most or first segment in the name is given more weight. When the SurnameAnchorSegment = "last", both comparand segments are right-aligned and the last segment in the name is given more weight. Thus, if only two surname segments exist, then the right-most or last segment is given more weight if SurnameAnchorSegment = "last".

The way in which the SurnameAnchorSegment gives more weight to certain name segments is through the use of the SurnameAnchorFactor. For example, if the SurnameAnchorSegment = "first", but neither of the similar digraph names are in the first position, then the SurnameSegmentScore is multiplied by the SurnameAnchorFactor.

```
SurnameAnchorSegment : first
SurnameOutOfPositionFactor : .65
SurnameMode : highest

SurnameAnchorFactor : .70

Query : Lopez Garcia, Luis
Candidate : Santos Garcia, Luis
```

In this example, since the SurnameAnchorSegment = first, the two comparands are left-aligned. After left-aligning the comparands, more weight should be given to the name in the first position. The name "Garcia" matches; however it is in the last position in both the query and the candidate (i.e., it is <u>not</u> in the first position, which is the SurnameAnchorSegment). Therefore, the SurnameSegmentScore(1.00) is multiplied by the SurnameAnchorFactor(.70), thus yielding a score of : SurnameSegmentScore * SurnameAnchorFactor = 1.00 * .70 = .70. Because the SurnameMode = "highest", the SurnameScore = the highest SurnameSegmentScore (1.0). Therefore, the way in which one surname is given more weight is actually to devalue the other or give it less weight.

If the same parameter settings are in effect, and the segments in the first position in both comparands are being compared, then the SurnameSegmentScore is not devalued. For example:

```
SurnameAnchorSegment : first
SurnameOutOfPositionFactor : .60
SurnameMode : highest

SurnameAnchorFactor : .70
```

Query : Gonzalez Garcia, Mario
Candidate : Gonzalez Salvador, Mario

In this case, after left-alignment, the name "Gonzalez" is considered to be in the first position in both comparands, and since the SurnameAnchorSegment ="first", it receives a SurnameSegmentScore of 1.00.

Similarly, if the SurnameAnchorSegment = "last", the emphasis is upon the last element in the Surname. In the following example, after right-alignment, the names "Lopez" and "Santos" do not share any common digraphs, and therefore receive a SurnameSegmentScore of 0. However, the name "Garcia" matches and is in the second position in comparands, therefore it receives a score of 1.00 and is not multiplied by the SurnameAnchorFactor since the SurnameAnchorSegment = "last". Since the SurnameMode = "highest", the SurnameScore = 1.00, which is the highest SurnameSegmentScore.

SurnameAnchorSegment : last
SurnameOutOfPositionFactor : .65
SurnameMode : highest
SurnameAnchorFactor : .70

Query : Lopez Garcia, Luis
Candidate : Santos Garcia, Luis

In contrast, if the evaluated segments, (i.e., Lopez and Santos in the example above) are not in the last position, the SurnameSegmentScore will be multiplied by the SurnameAnchorFactor.

SurnameAnchorSegment : last

Query : Gomez Hernandez, Mario
Candidate : Gomez Lopes, Mario

Although the name "Gomez" is an exact match, it is not in the last position in either comparand, and the SurnameAnchorSegment = "last". Therefore, "Gomez" is multiplied by the SurnameAnchorFactor. The SurnameSegmentScore(1.00) is multiplied by the SurnameAnchorFactor (.70) producing the SurnameSegmentScore = (.70).

The value of the SurnameAnchorSegment determines which of the name segments, if any, is to receive the most weight. Raising the SurnameAnchorFactor will actually give more value to the segment that is not the SurnameAnchorSegment, while lowering the SurnameAnchorFactor will lower the value of the segment that is not the SurnameAnchorSegment.

SurnameAnchorSegment
Possible Settings: {first, last, none}
Average Range: {first, last, none}
Default: {none}

SurnameAnchorFactor
Possible Settings: {0.00, 0.01,... 1.00}
Average Settings: {.50...70}
Default: {.70}

8.7 SurnameCheckTAQ

When the SurnameCheckTAQ = "off", no TAQ processing will take place at all.

When the SurnameCheckTAQ = "remove", then TAQ(s) will simply be removed from the name data.

When the SurnameCheckTAQ = "score", TAQ(s) will be identified, removed, and associated with each relevant name segment during preprocessing, and then the SurnameTAQDeleteFactor, SurnameTAQDeleteAbsentFactor, SurnameTAQDisregardFactor, and SurnameTAQDisregardAbsentFactor, will be multiplied against the SurnameSegmentScore, which will in effect reduce the value of the SurnameSegmentScore.

SurnameCheckTAQ
Possible Settings: {off, remove, score}
Default: {score}

8.8 SurnameMode (previously known as SNMODE)

The SurnameMode can be set to "highest", "average", or "lowest" depending upon how flexible or stringent one wants the parameters to be. "Highest" is the most flexible mode setting and "lowest" is the most stringent setting. SurnameMode only has an effect if there is more than one name in the surname. If there is more than one name in the surname, and the SurnameMode = "highest", then the SurnameScore will be set to the highest SurnameSegmentScore.

SurnameMode : highest

Query : Lopez Garcia, Maria
Candidate: Lopez Gonzalez, Maria

In this example, the highest SurnameSegmentScore will be used to evaluate the surname :"Lopez

Gonzalez". "Lopez" in the candidate matches "Lopez" in the query exactly, thus receiving a score of 1.00. Since "Gonzalez" has very few digraph matches with "Garcia", the SurnameScore will be set to 1.00, which is the highest SurnameSegmentScore.

If the SurnameMode = "average", the SurnameScore will be set to the average of the SurnameSegmentScores.

SurnameMode : average

Query : Lopez Garcia, Maria
Candidate: Lopez Gonzalez, Maria

The candidate segment "Lopez" matches the query segment "Lopez" with a score of 1.00. However, there is only one digraph match between "Garcia" and "Gonzalez", yielding a score of 1/9 or .11. Since SurnameMode = "average", the SurnameScore will be set to the average of the two SurnameSegmentScores. The average of the two SurnameSegmentScores in this example is (1+.11)/2 = .56.

If the SurnameMode = "lowest", and if there is more than one name in the surname, then the SurnameScore will be set to the lowest SurnameSegmentScore. In the example illustrated above, the SurnameScore will be set to .11. Clearly, SurnameMode = "lowest" is the most stringent setting.

If a threshold is defined to identify "hits", then setting the SurnameMode to "highest" will result in the return of more hits. Raising the SurnameMode to "average" will decrease the number of hits returned since the average score of the surnames must also pass the threshold. Raising the SurnameMode to "lowest" will further decrease the number of hits returned since both names must pass the threshold.

SurnameMode:
Possible Settings: {highest, average, lowest}
Average Range: {highest, average, lowest}
Default: {average}

8.9 SurnameExactInitialMatchScore

If SurnameCheckInitial is set to True, then the SurnameExactInitialMatchScore is used to indicate whether two single characters that match one another should be considered "exact matches", and therefore be assigned a score of 1.0. In some cases, it may be desirable to not consider two single characters as an exact match since it is possible that the two characters may represent two different names. In these cases, one might want to set the SurnameExactInitialMatchScore = (1-SurnameInitialScore)/2.

SurnameExactInitialMatchScore
Possible Settings: {0.00, 0.1, ... 1.00}

Average Settings: {1.0}
Default: {1.0}

8.10 SurnameInitialScore

The SurnameInitialScore behaves in the same manner as the GivenNameInitialScore but it applies to surnames rather than given names. This parameter will be useful in dealing with Hispanic surnames which frequently use an initial to represent High Frequency second surnames.

SurnameInitialScore
Possible Settings: {0.00, 0.1, ... 1.00}
Average Settings: {.60....90}
Default: {.85}

8.11 SNV-SCORE

The SNV-SCORE is the value given to a pair of Surname variants found in the SURNAME-VARIANT Table. The SNV-SCORE is generally set very high, usually at .95.

SNV-SCORE
Possible Settings: {0.00, 0.01, ... 1.00}
Default: {defined by variant pair}

8.12 SurnameOutOfPositionFactor, SurnameAnchorSegment (previously known as SNOOPS, ANCHSEG)

In order to determine the relative position of name segments in both the query and candidate, the tool shall establish an "index" of a segment based on the SurnameAnchorSegment. For SurnameAnchorSegment = “none” or “first”, the tool shall left-align the name segments. For SurnameAnchorSegment = “last”, the tool shall right-align the name segments.

The SurnameOutOfPositionFactor factor only applies to name segments that are out of position (i.e., not in the same relative position). When a surname segment is out of position, the SurnameSegmentScore is multiplied by the SurnameOutOfPositionFactor factor. In the following example, after left-aligning, the candidate name segments "Garcia" and “Gonzalez" are both considered to be out of position.

SurnameMode : average
SurnameOutOfPositionFactor : .65
SurnameAnchorSegment : first

Query : Gacria Gonzalez, Mario
Candidate : Gonzalez Garcia, Mario

The name "Gonzalez" is an exact match but it is out of position. Therefore, it receives a value of .65 (SurnameOutOfPositionFactor) X 1.00 (SurnameSegmentScore) = .65.

The name "Garcia" is not an exact match. Of the fourteen digraphs, there are 4 digraph matches.

```
The total possible digraphs are:

#G Ga ar rc ci ia a#
#G Ga ac cr ri ia a#

The matched digraphs are:

#G Ga ia a#
#G Ga ia a#
```

Therefore, the name receives a score of .65 (SurnameOutOfPositionFactor) X .57 (SurnameSegmentScore = 8/14 = .57).

The SurnameMode in this example = "average". Therefore, the SurnameScore = the average of the two SurnameSegmentScores, which is.51 = ((.65+.51)/2).

8.12.1 SurnameOutOfPositionFactor With Surnames Containing Only 1 Name Segment

In the following example, the name "Sanchez" is considered to be out of position.

```
SurnameAnchorSegment : first

Query :  Ramirez Sanchez, Luis
Candidate :  Sanchez, Luis
```

In the query "Sanchez" is considered to be in the last position, whereas, in the candidate, "Sanchez' is considered to be in the first position. Therefore, the SurnameScore = SurnameOutOfPositionFactor multiplied by the SurnameSegmentScore (1.00).

If the SurnameAnchorSegment = "last", then Sanchez would be considered to be in position, and the SurnameOutOfPositionFactor would not be applied.

If a NameThreshold is defined, raising the SurnameOutOfPositionFactor will generally result in the return of more names and lowering the SurnameOutOfPositionFactor will make it more difficult for names to pass the threshold.

SurnameOutOfPositionFactor:
Possible Settings: {0.00, 0.01,... 1.00}
Average Range: {.50...70}
Default: {.60}

8.13 SurnameTAQDisregardAbsentFactor

absent Surname Disregard TAQ score – refer to section on TAQ scoring in main document for description.

SurnameTAQDisregardAbsentFactor
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.80}

8.14 SurnameTAQDeleteAbsentFactor

absent Surname Delete TAQ score – refer to section on TAQ scoring in main document for description.

SurnameTAQDeleteAbsentFactor
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.90}

8.15 SurnameTAQDeleteFactor

delete Surname TAQ score – refer to section on TAQ scoring in main document for description.

SurnameTAQDeleteFactor
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.85}

8.16 SurnameTAQDisregardFactor

disregard Surname TAQ score – refer to section on TAQ scoring in main document for description.

SurnameTAQDisregardFactor
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.7}

8.17 LastNameUnknownScore

If one of the comparands has been identified as having “last name unknown”, then the segment score assigned when comparing that comparand with another is the LastNameUnknownScore.

LastNameUnknownScore
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.6}

8.18 NoLastNameScore

If one of the comparands has been identified as having "no last name", then the segment score assigned when comparing that comparand with another is the NoLastNameScore.

NoLastNameScore
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.65}

8.19 SurnameCompressedScore

In some instances, TAQ values become conjoined with stems in unpredictable ways. In some instances, two surname comparands are exact matches except for spacing (e.g., "de la Garcia" and "delaGarcia"). If this is determined to be the case, the tool will assign the SurnameCompressedScore to the SurnameScore.

SurnameCompressedScore
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.9}

8.20 SurnameThreshold (previously known as SNTHRESH)

The SurnameThreshold is the threshold which the SurnameScore must exceed in order for the candidate name to be included in the Results list. If a developer wants to define a threshold rather than return the TOP X names, then this parameter may be set to some value other than 0. Setting the SurnameThreshold to 0 essentially turns off the SurnameThreshold. As the SurnameThreshold is raised, fewer candidate names will be returned as it will be more difficult for a candidate name to pass the higher SurnameThreshold. Conversely, as the SurnameThreshold is lowered, more candidate names will be returned as it will be easier for a candidate name to pass the lower SurnameThreshold.

SurnameThreshold
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.50}

8.21 SurnameWeight

The SurnameWeight is the factor (weight) that can be applied to the SurnameScore when determining whether a candidate name is to be included in the Results list. This weight factor enables one to assign more or less emphasis to a potential candidate based on the SurnameScore.

The higher the SurnameWeight, the greater the value of the SurnameScore contribution to the overall NameScore. If the SurnameWeight is set to 0, the SurnameScore will not contribute any value to the overall NameScore. In Version 1, the exception to this occurs if the GivenNameWeight is also set to 0, in which case, the weight factors cancel one another out. In Version 1, we multiply the SurnameScore by the SurnameWeight as part of the default overall NameScore calculation. Note that developers may or may not choose to apply the SurnameWeight when calculating an overall NameScore if they create a different scoring algorithm.

SurnameWeight
Possible Settings: {0.0, 0.1, ...1.0}
Default: {1.0}

8.22 GivenNameCheckInitial (previously known as ISGNINITL)

The GivenNameCheckInitial behaves the same as SurnameCheckInitial, but applies to given names rather than surnames.

GivenNameCheckInitial
Possible Settings: {T, F}
Default: {T}

8.23 GivenNameCheckVariant (previously known as CHKVARIANT)

The GivenNameCheckVariant behaves in the same manner as the SurnameCheckVariant, but applies to given names rather than surnames. When SurnameCheckVariant = "T", a table containing GN Variants is referenced during the evaluation as well.

GivenNameCheckVariant
Possible Settings: {T, F}
Default: {T}

8.24 GivenNameCheckBias

This parameter behaves in the same manner as the SurnameCheckBias but it applies to given names rather than surnames.

GivenNameCheckBias
Possible Settings: {T,F}
Default: {F}

8.25 GivenNameCheckUnknownNotExist, NoFirstNameScore, FirstNameUnknownScore

GivenNameCheckUnknownNotExist is similar to SurnameCheckUnknownNotExist except that it applies to the GN field. The parameters for GivenNameCheckUnknownNotExist are also specific to the GN field.

GivenNameCheckUnknownNotExist
Possible Settings: {T, F}
Default: {F}

NoFirstNameScore
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.80}

FirstNameUnknownScore
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.85}

8.26 GivenNameCheckCompressed, GivenNameCompressedScore

GivenNameCheckCompressed
Possible Settings: {T,F}
Default: {F}

GivenNameCompressedScore
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.9}

8.27 GivenNameAnchorSegment, GivenNameAnchorFactor

The GivenNameAnchorSegment behaves in the same manner as the SurnameAnchorSegment but it applies to given names rather than surnames.

The GivenNameAnchorFactor behaves in the same manner as the SurnameAnchorFactor but it applies to given names rather than surnames.

GivenNameAnchorSegment
Possible Settings: {first, last, none}
Average Range: {first, last, none}
Default: {none}

GivenNameAnchorFactor
Possible Settings: {0.00, 0.01,... 1.00}
Average Settings: {.50...70}
Default: {.70}

8.28 GivenNameCheckTAQ

When the GivenNameCheckTAQ = "off", no TAQ processing will take place at all.

When the GivenNameCheckTAQ = “remove”, then TAQ(s) will simply be removed from the name data.

When the GivenNameCheckTAQ = “score”, TAQ(s) will be identified, removed, and associated with each relevant name segment during preprocessing, and then the GivenNameTAQDeleteFactor, GivenNameTAQDeleteAbsentFactor, GivenNameTAQDisregardFactor, and GivenNameTAQDisregardAbsentFactor, will be multiplied against the GivenNameSegmentScore, which will in effect reduce the value of the GivenNameSegmentScore.

GivenNameCheckTAQ
Possible Settings: {off, remove, score}
Default: {score}

8.29 GivenNameMode

GivenNameMode operates exactly the same way as the SurnameMode but it applies to given names rather than surnames.

GivenNameMode:
Possible Settings: {highest, average, lowest}
Average Range: {highest, average, lowest}
Default: {average}

8.30 GivenNameExactInitialMatchScore

If GivenNameCheckInitial is set to True, then the GivenNameExactInitialMatchScore is used to indicate whether two single characters that match one another should be considered “exact matches”, and therefore be assigned a score of 1.0. In some cases, it may be desirable to not consider two single characters as an exact match since it is possible that the two characters may represent two different names. In these cases, one might want to set the GivenNameExactInitialMatchScore = (1-GivenNameInitialScore)/2.

GivenNameExactInitialMatchScore
Possible Settings: {0.00, 0.1, ... 1.00}
Average Settings: {1.0}
Default: {1.0}

8.31 GivenNameInitialScore

The GivenNameInitialScore deals with the treatment of initials during a name check. In the following example, the initial "M" in the candidate could correspond to the name "Mohamed" in the query. Instead of considering it as a single digraph match, which in this case, would yield a score of .125, the "M" is given the value of the GivenNameInitialScore.

GivenNameInitialScore : .85 Query : Ali, Mohamed Candidate : Ali, M

If a NameThreshold is defined, raising the GivenNameInitialScore will result in the return of more good hits since the value of an initial in a potential hit has been raised. Likewise, lowering the GivenNameInitialScore will result in a decrease in the number of hits returned.

GivenNameInitialScore
Possible Settings: {0.00, 0.01, ... 1.00}
Average Settings: {.60....90}
Default: {.85}

8.32 GNV-SCORE

The GNV-SCORE is the value given to a pair of Given Name variants found in the GIVEN-NAME-VARIANT Table. The GNV-SCORE is generally set very high, usually at .95.

GNV-SCORE
Possible Settings: {0.00, 0.01, ... 1.00}
Default: {defined by variant pair}

8.33 GivenNameOutOfPositionFactor (previously known as GNOOPS)

The GivenNameOutOfPositionFactor factor operates in the same manner as the SurnameOutOfPositionFactor.

GivenNameOutOfPositionFactor:
Possible Settings: {0.00, 0.01,... 1.00}
Average Range: {.50...70}
Default: {.55}

8.34 GivenNameTAQDisregardAbsentFactor

absent GN Disregard TAQ score – refer to section on TAQ scoring in main document for description.

GivenNameTAQDisregardAbsentFactor
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.80}

8.35 GivenNameTAQDeleteAbsentFactor

absent GN Delete TAQ score – refer to section on TAQ scoring in main document for description.

GivenNameTAQDeleteAbsentFactor
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.90}

8.36 GivenNameTAQDeleteFactor

delete GN TAQ score – refer to section on TAQ scoring in main document for description.

GivenNameTAQDeleteFactor
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.85}

8.37 GivenNameTAQDisregardFactor

disregard GN TAQ score – refer to section on TAQ scoring in main document for description.

GivenNameTAQDisregardFactor
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.7}

8.38 FirstNameUnknownScore

If one of the comparands has been identified as having "first name unknown", then the segment score assigned when comparing that comparand with another is the FirstNameUnknownScore.

FirstNameUnknownScore
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.6}

8.39 NoFirstNameScore

If one of the comparands has been identified as having "no first name", then the segment score assigned when comparing that comparand with another is the NoFirstNameScore.

NoFirstNameScore
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.65}

8.40 GivenNameCompressedScore

In some instances, TAQ values become conjoined with stems in unpredictable ways. In some instances, two given name comparands are exact matches except for spacing (e.g., "nur al din" and "nuraldin"). If this is determined to be the case, the tool will assign the GivenNameCompressedScore to the GivenNameScore.

GivenNameCompressedScore
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.9}

8.41 GivenNameThreshold (previously known as GNTHRESH)

The GivenNameThreshold is the threshold which the GivenNameScore must exceed in order for the candidate name to be included in the Results list. If a developer wants to define a threshold rather than return the TOP X names, then this parameter may be set to some value other than 0. Setting the GivenNameThreshold to 0 essentially turns off the GivenNameThreshold. As the GivenNameThreshold is raised, fewer candidate names will be returned as it will be more difficult for a candidate name to pass the higher GivenNameThreshold. Conversely, as the GivenNameThreshold is lowered, more candidate names will be returned as it will be easier for a candidate name to pass the lower GivenNameThreshold.

GivenNameThreshold
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.50}

8.42 GivenNameWeight

The GivenNameWeight is the factor (weight) that can be applied to the GivenNameScore when determining whether a candidate name is to be included in the Results list. This weight factor enables one to assign more or less emphasis to a potential candidate based on the GivenNameScore. The higher the GivenNameWeight, the greater the value of the GivenNameScore contribution to the overall NameScore. If the GivenNameWeight is set to 0, the GivenNameScore will not contribute any value to the overall NameScore. In Version 1, the exception to this occurs if the SurnameWeight is also set to 0, in which case, the weight factors cancel one another out. In Version 1, we multiply the GivenNameScore by the GivenNameWeight as part of the default overall NameScore calculation. Note that developers may or may not choose to apply the GivenNameWeight when calculating an overall NameScore if they create a different scoring algorithm.

GivenNameWeight
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.80}

8.43 NameThreshold

The NameThreshold is the threshold which the NameScore must exceed in order for the candidate name to be included in the Results list. If a developer wants to define a threshold rather than return the TOP X names, then this parameter may be set to some value other than 0. Setting the NameThreshold to 0 essentially turns off the NameThreshold. As the NameThreshold is raised, fewer candidate names will be returned as it will be more difficult for a candidate name to pass the higher NameThreshold. Conversely, as the NameThreshold is lowered, more candidate names will be returned as it will be easier for a candidate name to pass the lower NameThreshold.

NameThreshold
Possible Settings: {0.0, 0.1, ...1.0}
Default: {.60}